Understanding the mobile game maintenance process based on the version history data

Mengyuan Yang

University of Tampere
Faculty of Natural Sciences
Degree Programme in Computer Sciences
M. Sc. thesis
Supervisor: Zheying Zhang
December 2017

University of Tampere
Faculty of Natural Sciences
Degree Programme in Computer Sciences
Mengyuan Yang: Understanding the mobile game maintenance process based on the version history data
M.Sc. thesis, 66 pages, index 8 and appendix 21 pages
December 2017

Nowadays, a vast number of mobile games appear in the 'App Store'. The fierce competition pushes the suppliers updating their games continuously. Therefore, understanding the maintenance process of mobile games could provide useful guidelines to plan and conduct the application maintenance.

This research applies two data mining methods to analyze the mobile application version history data in the game category, which records all the versions of each mobile games. Firstly, the data is classified to different types of maintenance. The proportion of maintenance types illustrates the critical maintenance activities of mobile games. Secondly, data clustering is used to group the mobile games which have similar update trends.

The data from text classification phase implies that, most of the game maintenance activities are related to adding new features, correcting bugs, improving performance, and optimizing game size and contents. From the result of data clustering, there are four stages could be found from the data. At the first stage, mobile games start with different updating trends. Most of games releases new versions randomly. The dominate maintenance activities is enhancive maintenance. At the second stage, most of games release new features in a relatively high frequency and volume. The dominate maintenance activities is enhancive maintenance as well. At the third stage, mobile games update keeps a high frequency manner as well, however, the volume can be differ. The proportion of other maintenance types start to increase. At the final stage, the release frequency and volume keep in a stable state or gradually decrease. The proportion of other maintenance types continue growing, especially in corrective maintenance and performance.

Key words and terms: software maintenance, text mining, data clustering, mobile game maintenance, mobile version history data.

ACKNOWLEDGEMENTS

Two years ago, I started my journey of writing the thesis. It witnessed how the knowledge related to this thesis being learned, known and acquired. It was not easy to collect and analyze the version history data. Many methods were tried and discarded, but finally I have this thesis finished. I would like to express my sincere thanks for all the people who have helped last few years.

Foremost, I want to thank my supervisor, Dr.Zhang who took care every detail of my thesis from the choosing the topic to checking the grammars. If without her help, I can not go that longer.

I would like to thank Xiaozhou Li. His great idea about application version history data does inspire me. He also gave me many useful pieces of advice and comments on my thesis.

My thank also goes to my dear friends Dan Yang, Zhaodong Fan and Menglin Xu. Thanks for your kindly accompany when I am out of energy.

Finally, I would like to express my special thanks to my family, especially my parents, whose unconditional love is the biggest motivation for me to continue study in Finland.

Tampere, June 6th, 2017

Mengyuan Yang

CONTENTS

List of Figures

List of Tables

ABBREVIATIONS AND SYMBOLS

NB	Naive Bayes
KNN	K-Nearest Neighbourhood
DT	Decision Tree
SVM	Support Vector Machine
HAC	Hierarchical Clustering
DTW	Dynamic Time Warping
#	The number of

1 INTRODUCTION

1.1 Motivations

Together with the popularity of smartphones and mobile devices, the mobile applications business has been blooming in an exponential scale. Nowadays, the mobile application market has become one of the most active and profitable markets in the world. A report of Statista [2017] shows that there are currently about 2.2 million applications available on the iOS platform, where as many as 60 thousand applications are added per month with this rate itself growing.

A huge number of applications lead to fierce competition. In order to improve the competitiveness and serve users with better experiences, application providers update and maintain their applications continuously. For example, Facebook[1] has more than 70 releases in the past six years since it has been launched. Several similar cases, such as Candy Crush Saga[2], Clash of Clans[3] can also show that the maintenance of mobile applications is of particular importance for application providers. In practice, however, such an important process often suffers from the lack of accurate information, useful guidances or supporting concepts[Basili et al., 1996]. Even though the concepts and methods of software maintenance can be applied in mobile application maintenance, the mobile applications have their unique characteristics. Therefore, understanding the maintenance process in the domain is essential to further develop guidelines for mobile application maintenance.

Currently, magnanimity data related to applications has been produced in 'App Store' and 'Google play', such as the application rank, user ratings and download numbers. Those data contain valuable information for exploring user behaviors, marketing trends and other interesting issues. For example, Chen and Liu [2011] found the correlation between the application price and the application rank. Pagano and Maalej [2013] illustrate the correlation between the length of the user review and the application price. Moreover, a method called AR-miner, to extract user requirements from user reviews, is proposed by Chen et al. [2014]. Those studies mainly focus on the data as user reviews to analyze how useful they are. In addition, limited studies explored application version history data. Only Li et al. [2014] proposed three mobile application process models by observing the version history data of 100 mobile applications in different application categories in 'App Store'.

[1] https://itunes.apple.com/us/app/id284882215?mt=8
[2] https://itunes.apple.com/us/app/id553834731?mt=8
[3] https://itunes.apple.com/us/app/id529479190?mt=8

Mobile application version history data is the data which contains full records of when(the version release date) an application is updated and what(the version description) is updated. Moreover, all of them are stored in a textual format. It starts with when the application is launched on the 'App Store', and ends with when the application is off the shelf. Li's research has demonstrated that the mobile application version history data contains valuable information in exploring mobile application maintenance issues.

Different from Li's research, this research studies version history data from 238 popular mobile applications in the game category. All the data are collected from the iOS platform and analyzed through two data mining steps: text classification and data clustering. The research is aiming at summarizing the maintenance process of mobile game applications. Results from data mining can help mobile game application maintainers and decision makers in planning maintenance activities.

1.2 Research question

Software maintenance is a complicated and time-consuming process in the whole software life cycle[Lientz et al., 1978]. In terms of the complexity, there are many maintenance types corresponding to variety modification requests from users. For example, an application maintainer categorizes a 'bug fix' report as as corrective maintenance. Moreover, many factors can impact the software maintenance process, including the opinions of users, the business plan of software company and the knowledge of maintainers, etc.. Considering the cost, there are many necessary activities to be done during maintenance phase, such as prioritizing the importance of user requests, implementing and reviewing changes. Furthermore, with the number of modifications increasing, both the complexity and the cost will also increase.

The mobile application maintenance seems more brisk, when planning one or two releases is not enough for demanding users. An increasing number of application suppliers are obliged to prepare several releases in advance, which requires them to design the maintenance process and to plan maintenance activities at the early stage or even during the application development phase[Bourque and Farley, 2014]. Furthermore, the traditional concepts of software maintenance can not fit properly to the fast changing marketing trend and the technology evolution[Bennett and Rajlich, 2000].

No matter for which purpose the application is developed, the application provider always expect a long application lifespan and the sustainable staying in business[Bourque and

Farley, 2014]. On the business aspect, applications with a longer time span have more impact on the market. Continuous and well-scheduled maintenance is a critical method to achieve the purpose. Therefore, this research addresses one research question: How does mobile application(game) maintenance evolve over time? Answers to the question could provide an insight of mobile game maintenance process concerning the trending ways of maintenance planning.

Usually, there are many different application categories in 'App Store' such as game, finance, health, etc.. Those categories support users with a convenient application navigation. Due to the time limitation, only applications in the game category will be explored in the thesis.

1.3 Research methods

Before answering the research question, the concepts and processes related to software maintenance, mobile application maintenance are necessary to be introduced as the background knowledge, so as to the data mining methods used in this thesis.

For most mobile game providers, it is promising to have a regular update schedule or strategy, which can help them arrange the maintenance activities and settle the releasing times. The updating trend, which is depicted by the number and the releasing date of updates, can be a useful reference for designing suitable application maintenance strategies. Therefore, based on the mobile application version history data, the research question can be answered from two aspects: the version update content and the trend of application maintenance amount changing with time.

Answering this question from the aspects mentioned above requires a well-designed data mining process. Firstly, due to the application version history data is not readily available, it has to be collected from two website sources. Afterwards, understanding the content of version descriptions is necessary. Usually, version descriptions are connected tightly with different maintenance types. For example, 'Bug fix' is corresponding to a corrective maintenance, 'We add some new features.' is an enhancive maintenance. Considering the huge workload for categorizing all the version descriptions manually, classifying them by machine is more effective. That is why text classification is used. Subsequently, a trend curve for each application will be depicted with the version release time as the x-aixs and the total numbers of the maintenance types as the y-aixs. Finally, applying data clustering method on those curves could extract a common practice of mobile game maintenance.

1.4 Thesis outline

This thesis is organized in seven chapters. Chapter 2 covers the contents about software maintenance concepts, processes and maintenance types. Chapter 3 discusses mobile application maintenance and mobile version history data. Chapter 4 addresses processes and methods of data mining which will be used for the application history data collection and analysis. Chapter 5 explains the concrete processes of analyzing application version history data. Chapter 6 illustrates the results of text classification and data clustering. Chapter 7 concludes this thesis with the discussion of limitations and recommendation for future research.

2 SOFTWARE MAINTENANCE

Software maintenance, as an important phase of software development lifecycle, consumes 75-80 percent of total system resources[Lientz et al., 1978]. Since Lehman [1969] proposed an empirical study, improving company's programming effectiveness, software maintenance has become a prolific field of research[Herraiz et al., 2013]. With the development of software functions, platforms and technologies, the concept of software maintenance continues to grow[Zvegintzov and Parikh, 2005].

In IEEE standard, software maintenance is described as 'modification of a software product after delivery to correct faults, to improve the performance or other attributes, or to adapt the product to a varied environment'[Mamone, 1994]. Another definition from ISO/IEC describes 'software maintenance as the totality of activities required to provide cost-effective support to a software system. Those activities are performed in two stages. The first stage in which the software is under development or testing is called the pre-delivery stage. The second occurs when the software product is delivered to users, and it is called post-delivery stage. The pre-delivery stage activities include planning for post-delivery operations, maintainability, and logistics determination for transition activities. The post-delivery activities involve software modification, training and operating or interfacing to a help desk'.[Standard, 2006]

The second definition by ISO/IEC covers all the maintenance activities described in the first definition. Besides, it does not limit the modification activities during software post-delivery stage. It also emphasizes the importance of maintenance tasks planning through the whole software product life. Therefore, the following discussion of maintenance is on the basis of this definition.

2.1 Software maintenance objectives

From the perspective of software users, software needs to keep running smoothly. System faults or bugs are zero-tolerance for users. With time goes by, more changes will be proposed by users, therefore satisfying users with new functions and features forms an important objective of maintenance. Besides, when the usage environment changes, the changes include hardware replacement, operating system update, and related government policy evolution, can trigger an adjustable software maintenance. Sometimes, software providers will support users by including training and consulting service in software

maintenance.[Grubb and Takang, 2003]

In the providers' point of view, maintenance tasks also need to be done to facilitate future maintenance work. It includes such activities as code restructuring, documentation updating and so on. Furthermore, maintenance can preserve the integrity of the software, extend software lifespan and make more profit from market[Bourque and Farley, 2014].

2.2 Software maintenance process

A software maintenance process contains activities and tasks necessary to modify an existing software product while preserving its integrity[Goel, 2011]. In small software development groups, maintenance may follow an informal process. The development and maintenance tasks have not a clear boundary: a developer can be the maintainer and vice versa. On the contrary, in big software organizations, it is a structured process with documentation produced at each stage. However, at an abstract level, all maintenance processes have the same fundamental activities[Standard, 2006], as shown in Figure 2.1.

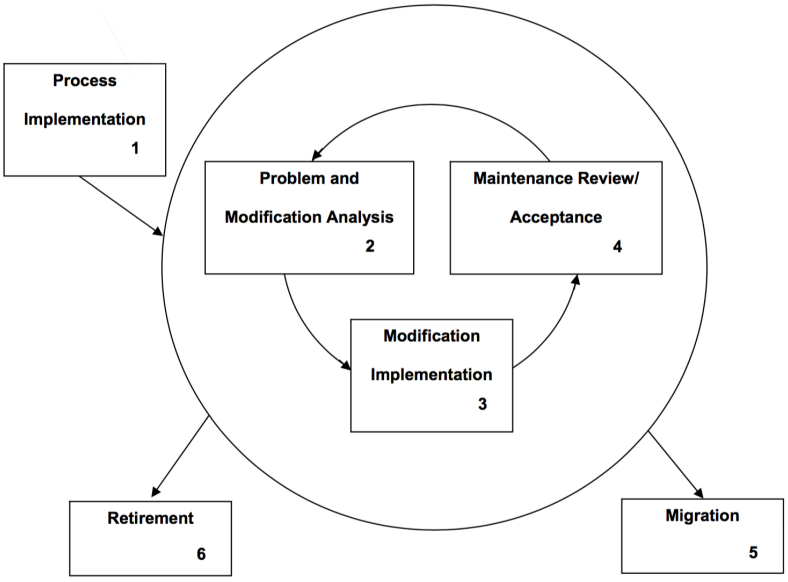

Figure 2.1. Software maintenance processes[Standard, 2006]

In Figure 2.1 each rectangle represent one of the six maintenance process phases. The number in the rectangle indicates the order of maintenance phases. The tree phases, prob-

lem and modification analysis, modification implementation, and maintenance review, connected in the inner circle illustrate activities directly related to software modification. The outer cycle describes the activities which are not directly connected with software modifications. It includes process implementation, software migration, and software retirement. The detailed explanations for each phase are introduced as follows.

During the process, implementation phase, many plans are prepared under different considerations. On the business level, maintenance budgetary and financial risks need to be carefully evaluated and managed. Regarding the transition level, clear interfaces and complete specifications can smooth work handover staffs from the development process to the maintenance process. About concrete software modifications, user requests and modifications need to be planned and managed in a traceable manner. Except that, all factors impact software maintenance should be considered and evaluated as earlier as possible. As ISO standard mentioned that the process implementation phase should start with software development simultaneously[Standard, 2006].

Once the software product is delivered to users, feedback contains modification requests and problems reports will trigger the problem and modification analysis phase. At first, maintainers need to analyze modification requests or problem reports. The analysis usually concerns the type of requests or the type of maintenance, e.g. classifying 'bug fixing' to corrective maintenance type; the scope of requests, the size of modification, the costs and time involved in modifying user requests; and the criticality of the requests, such as the impact on safety or security. Secondly, a reasonable prioritizing mechanism for requests and up-to-date documentation are also required in a manageable manner. Finally, maintainers and other stakeholders should obtain approvals for implementing the requests. The essential outcome of this phase is the approvals of all implementing modification.

While all stakeholders make a consensus on the modifications, those modifications will be implemented. During the implementation phase, maintainers develop and test the code or the function of the software product just as the software development process. However, the difference is software maintenance aims to modify the existing software system. The outcome of this process includes updated documentation, modified source codes, and modification test results.

Each time when modification happens, it needs to be reviewed after the implementation. Maintenance review/acceptance is a quality assurance phase for the modified software. It ensures that each modification is implemented by a correct method and matches approved standards from the last phase. If the modification fails to satisfy the user request, it needs

to start from problem analysis to maintenance review phase iteratively.

Occasionally, software needs to be migrated to a different environment. For example, migrating a iOS game to Android platform. To migrate a system to a new platform, the maintainer needs to tidy up all the related documents and arrange needed migration actions in advance.

When a software is not useful any more, it must be retired. Usually, before a software product is retired, an analysis should be conducted. It is helpful for making a suitable retirement decision. Theoretically, software is a combination of codes, it can never retire. However, for some economical reason, the retirement decisions must be made.

2.3 Software maintenance types

Software maintenance types support software maintainer with a clear guideline to organize and deliver maintenance tasks. Moreover, it is also a significant manner to analysis maintenance logs or maintenance contents. The earliest recorded maintenance typology is given by Swanson [1976]. It categorizes maintenance activities into three categories from the perspective of software uses. They are adaptive maintenance, perfective maintenance and corrective maintenance. Adaptive maintenance describes the activities of modifying the system to cope with software environmental change. Perfective maintenance is to implement new or changed user requirements which acts as functionality enhancement. Corrective maintenance refers to diagnose and fix bugs or system exceptions.

In addition, Harjani and Queille [1992] notice that non-programming-related features are also essential for software maintenance. They add user support activities as one kind of software maintenance activities. User support includes activities aiming at answering an explicit request for information from the user to correct his/her misunderstanding of the operation of a software. For example, software used for industrial production often offers users with training service.

With the improvement of software engineering concepts, researchers realize the significance of software maintainability. So preventive maintenance is added and recorded in IEEE standard of software maintenance in 1994. Preventive maintenance describes the activities in increasing software maintainability or reliability to prevent problems in the futureMamone [1994].

Furthermore, to emphasize the changes in software requirements, Kitchenham et al. [1999] classify software maintenance types into two main types: Corrections, meaning to correct a system defect or exception, and Enhancements, meaning to implement a change for the system. In the enhancements category, there are three subclasses including existing requirements change, new system requirements and the implementation enhancement. For example, redesigning the structure of the code to improve the software performance is an enhancement of implementation.

Despite of the previous typologies perfecting software maintenance types, the definition of types are different or overlapped with each other. Such a variety often causes misunderstanding. For example,the 'perfective maintenance' can also be subdivided. It concludes the contents such as improving code structure, adding a new function, even fixing errors in documentation. In practice, these activities must be discussed in different maintenance types and assigned to different departments. Moreover, a typology with no ambiguity is necessary for text classification, because some classification methods require different classes that are independent with each other.

Typology from Chapin et al. [2001] summarize different maintenance types proposed in previous studies, on the basis of a long-term practical observation and detection of maintained system. This typology has been used widely in discussing software maintenance issues. Buckley et al. [2005] address software changes through Chapin's typology. Research from Eden and Mens [2006] adapt this typology to count metrics for software maintenance. Wang et al. [2006] claim that in the classification of software maintenance, this topology can be used as a roadmap.

This typology classifies maintenance into four clusters based on three questions asked most frequently in real software maintenance tasks. Each cluster contains several maintenance types. The whole picture of typology is shown in Figure 2.2.

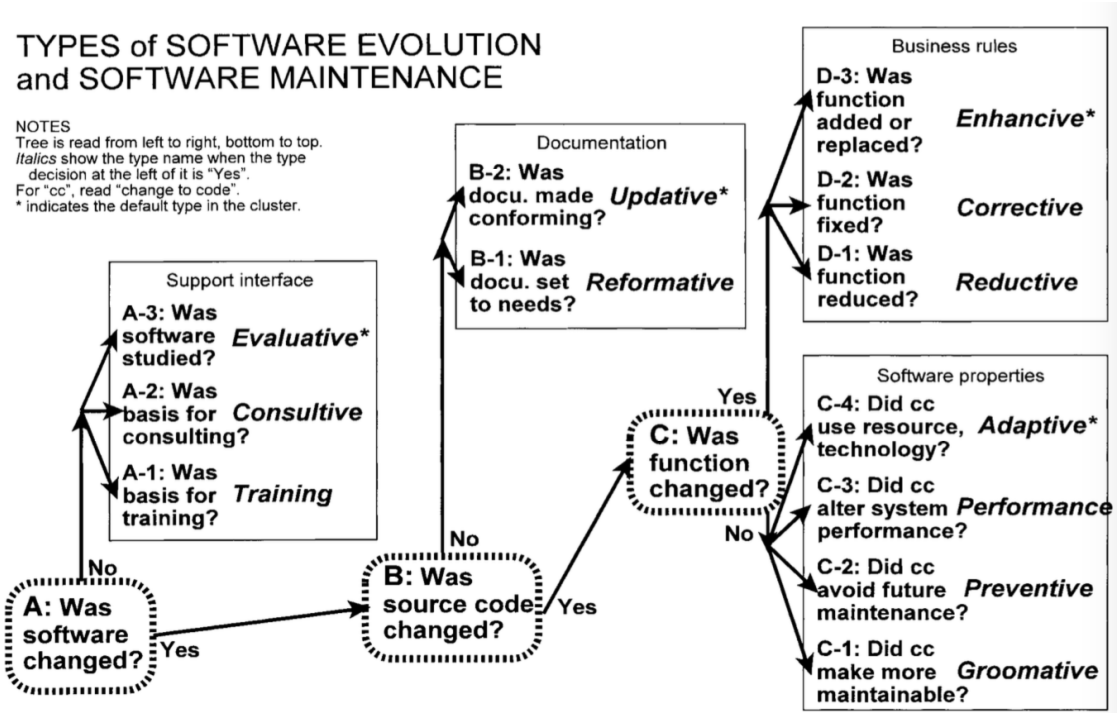

Figure 2.2. Decision tree for maintenance types Chapin et al. [2001]

The four solid-line-rectangles represent four clusters, i.e. support interface, documentation, business rules and software properties. The dotted line rectangles show questions used for clustering the maintenance activities. They are based on the evidence of software maintenance activity and correspond with questions frequently asked in software maintenance practice. The questions include: A - Was the software changed? B - Was the source code changed? C - Was the function changed? The questions are further elaborated into a set of sub-question as presented in each cluster. Following those questions, maintenance activities can route to the appropriate cluster and then to the suitable maintenance types.

The clusters are prioritized by their different impacts as Figure 2.3 shows. The impact on software is shown from the top to the bottom. The degree of impact is sorted in an ascending order. For example, training a user to understand more about the system (in the support interface cluster) may have lower effect than fixing a system bug(in the business rules). Because the former one does not change the code, but the later one does. The impact on the business process is shown from the left to the right. From the left to right, the degree of impact is an ascending order as well. The number of small cubes implies the influence of the business process. For instance, enhancing the software with new functions may attract more users than modifying name of test report.

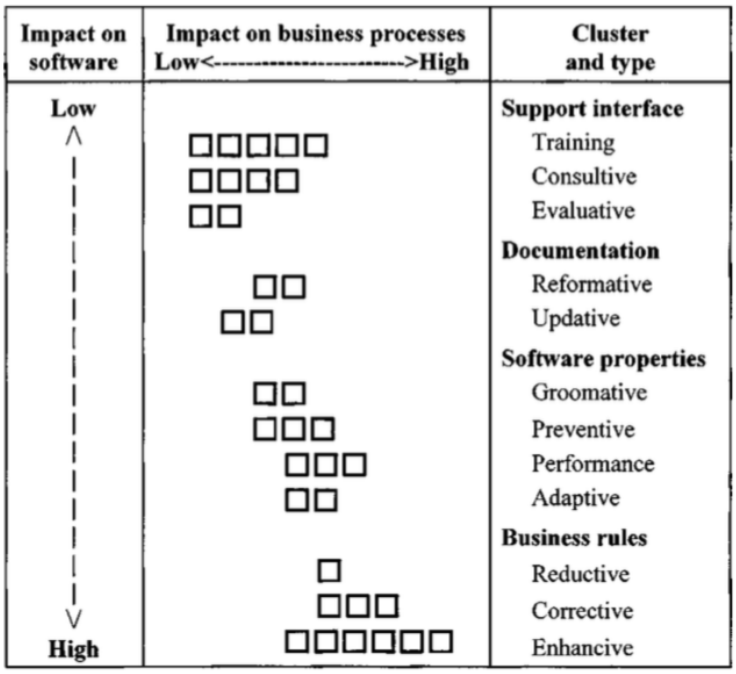

Figure 2.3. The importance table of each cluster[Chapin et al., 2001]

After going through the whole structure of the typology, the details is defined below. **Support interface** deals with non-programming-related activities. It follows the question route: 'Was software changed? No.'. For example, the software itself is not changed, but the training service is changed. It includes activities of training, consultative, and evaluative maintenance types. The training type includes conducting lessons and training users to fulfil system usage. Consultive maintenance happens commonly. Activities as answering questions at a help desk for customers is recorded in this type. Evaluative maintenance involves activities such as auditing, searching, examining, studying, diagnostic testing and those activities can help maintainer understand the software.

Documentation is a set of human readable text or graphics specifying or describing computer software(e.g. requirements specification, data tables and test plan and so forth.). In this cluster, it follows the question route: 'Was software changed? Yes. 'and was source code changed? No.' It describes the documentation update and modification. Documentation cluster consists of updative and reformative maintenance types. Updative maintenance is more closely to the system as implemented. The documentation needs to be updated when the modifications have happened. It involves such activities as replacing old non-code documentation with current documentation and adapting the change with other related documentation. Reformative maintenance reformulates or refines the non-code documentation but keep the code untouched, such as changing the fonts or paragraph

formats of a document.

Software properties include static or dynamic attributes of a system or its performance, such as system speed, system size and the implementation language. It concludes maintenance activities response in system non-functionality change but with source code change. For example, the code refactoring does change the code, but it does not change the functionality of the system. It follows the question route: 'Was software changed? Yes. Was source code changed? Yes. Was function changed? No.' This cluster includes adaptive maintenance, preventive maintenance, groomative maintenance, and performance. Groomative maintenance involves activities as doing recompilation, removing or updating codes more elegant ones, managing system versions and access authorizations. It mainly focuses on improving the communication between software maintainers and the environment. The preventative type concerns increasing maintainability to prevent problems in the future. Performance type involves such processes as optimizing user access time, reducing system response time and reducing system cache. Adaptive maintenance is to modify the system to deal with the software environment changes. For example, if the operating system changes, the software will be adapted for the new operating system.

Business rules are a step or set of steps in a process that mainly summarizes activities caused by functionality change. The maintenance types here are of the most frequent and most significant in software maintenance[Chapin et al., 2001]. This cluster follows the question route: 'Was software changed? Yes. Was source code changed? Yes. Was function changed? Yes.'. It includes enhancive maintenance, corrective maintenance and reductive maintenance. Enhancive maintenance is of particular significance for software maintenance. It implements changes and add new functionality to the system. For instance adding new themes, new play models on a mobile game. Corrective maintenance refers to handle exceptions, e.g. fixing crash bugs for an application. Reductive maintenance involves removing or replacing some functions or business rules from the implemented system. e.g. A mobile game supplier removes the Christmas theme from a game after Christmas.

2.4 Factors impact on software maintenance

The factors that impact on software maintenance process execution and maintenance planning is discussed in two stages. At the pre-delivery stage, maintenance tasks mainly focus on arranging and preparing resources for post-delivery stage maintenance works and help software process transmits smoothly from development process to the maintenance pro-

cess. The factors that impact those activities are summarized below.

- **Type of software** - Software maintenance varies from the type of software being maintained[Sommerville, 2004]. Stallman et al. [1998] classified software into free and nonfree software by the software copyright status. Even though there is no unified classification, the type of software can influence maintenance process in some degree. For example, maintaining desktop software as Microsoft Word requires more maintainers and time than updating a small mobile application developed by a small code group. In terms of mobile applications, a communication application as Skype tends to arrange more resource on 'reducing connected time' or ' fixing a bug of signals loss'. On the other hand, a game application, such as 'Clash of Clans' is willing to add more new features. Therefore, understanding the type of software can help software provider with clear target and focus.

- **Scale of software organization** - As mentioned before, in small software develop groups, maintenance is an agile but informal process. In big software organizations, software maintenance works with a structured process with prepared documentation in each stage. Different software organizations arrange resource differently. For example, a coder in small software development group, may experience the development and maintenance at the same time.[Bourque and Farley, 2014]

- **Technique support** - Technique support includes the training of maintainers and the tools used for software maintenance. For example, some useful version control tools can help maintainers release new versions or collect user reviews automatically.[Chen et al., 2014]

At the post-delivery stage, the maintenance activities focus on sustaining user satisfaction and keeping regular releases[Sneed and Brossler, 2003]. The factors that have more influence in this stage are listed below.

- **Objectives** - Each modification and release must have a clear objective. For example, if the objective is to attract users, enhancive maintenance updates will occupy a higher proportion in the release. Alternatively, if the objective is to keep the existing users, performance or corrective maintenance may occur in a high proportion. The objective is a mixture of different aspects which include business value, user intention, risks etc.. The trade-off between different aspects shall be taken into account.[Saliu and Ruhe, 2005].

- **User satisfaction** - User satisfaction is the main motivation of software providers to maintain their products or services. User feedbacks can illustrate the degree of user satisfaction and it contains rich resources for maintaining activities. For example, users can find errors which are not found during the testing phase.[Bourque and Farley, 2014].

- **Modification prioritization mechanism** - Modification prioritization mechanism is the manner of ordering the modifications. Because not all modifications can be achieved in one release, the prioritization mechanism is critical to make sure the most urgent or important modifications are achieved in releases. It is also tightly connected with software maintenance objectives and user satisfaction.[Saliu and Ruhe, 2005].

- **Time horizon** - The time horizon refers to the time interval required for releasing a new system version. There are two kinds of time releasing pattern. One is releasing with a fixed and pre-determined time interval for modification. The other is planning with flexible intervals. A fixed release cycles are commonly used in release planning.[Saliu and Ruhe, 2005]

3 MOBILE APPLICATIONS MAINTENANCE

Mobile application is the software which runs on mobile devices, smartphones and tablet computers. The same as the concept of software maintenance, mobile application maintenance is the totality of activities required to provide cost-effective support to a mobile application. It can be separated into two stages based on whether the application is launched on marketplace or not. The pre-launch stage which concludes activities planning for future maintenance tasks such as planning and analyzing financial, human resource and risks. The post-launch stage concludes activities such as software modification, technique supporting or release planning.

Most of the desktop software has been designed and implemented for a long term session usage, a relatively complete function set is delivered at an early stage [Saliu and Ruhe, 2005]. However, for mobile applications, launching it on the 'App Store' is just a beginning. The post-launch stage is crucial for the mobile application maintaining and growing. Therefore, scheduling maintenance content with the proper time interval is the main concern in mobile application maintenance.

3.1 Characteristics of mobile applications and mobile games maintenance

Even though the software maintenance methods can be applied mechanically in the mobile application maintenance process, mobile applications do have their unique characteristics which bring considerations from different perspective[Salmre, 2005].

The characteristics of mobile applications can be explained from several aspects[Rosado da Cruz, 2016]. Hardware is the foundation of any types of application, while the limitations of the hardware can inevitably affect the usability of mobile applications in terms of screen size, battery standby time and storage etc.. Moreover, the maintenance lifecycle of mobile application is shorter than that of desktop software. Because mobile applications are relatively small. Several thousand lines of source code in average, it is relatively easy for modifying[Wilson, 2017]. However, the advantage of mobile applications can not be ignored, as they can be used anytime anywhere. People can use a mobile application to deal with short and quick sessions such as calling, messaging or email checking.

Therefore, considering those characteristics mentioned above, the factors impact on mo-

bile application maintenance are listed as follow.

- **Keep releasing** - Comparing with desktop software, mobile application is relatively small, users can acquire functions or get tired of it in a short period. For example, some people does not know all the functions supported by 'Microsoft Word'. This situation may not happen to the mobile applications. The application provider shall keep releasing new versions to attract new users and keep existing users. [Greer and Ruhe, 2004]

- **Application size optimization** - Because of the limited storage and battery of mobile devices, application maintainers have to control the application size by restructuring codes or logics, and removing functions. For example, during Christmas, the application providers offer the application in a Christmas theme. After the holiday season, the theme will be removed.

- **Time sensitive** - Because people tends to do short sessions on mobile devices, mobile application users are time sensitive. For example, facing with a desktop software, waiting for 30 seconds for responding may be acceptable, but not for a mobile application. Performance problems such as slow starting time or poor network connection, low data transfer speed can greatly increase user dissatisfaction. Some corrective maintenance such as solving crushes problems shall be responded quickly too.

Mobile game is a special kind of mobile application. Nowadays, mobile games have become one of the best ways to allow users to spend spare time pleasurable. What makes mobile game such special? It is enjoyment. Research from Van der Heijden [2004] believe that perceived enjoyment plays a dominate role in keeping the willingness of user use the application. Which means that except the basic characteristics of mobile application mentioned before, perceived enjoyment is another key feature to be considered when maintaining a mobile game. Even through there is no a accurate way to measure the degree of enjoyment, however, research from [Desurvire et al., 2004; Korhonen and Koivisto, 2006; Jeong and Kim, 2007; Nacke, 2009], support some useful method could help, such as adding new content or story line, redesigning game rules and introducing social elements.

3.2 Mobile application maintenance process models

The phenomenon of mobile industry blooming has proved that a well-defined process model is needed for mobile application maintainers to maintain their applications in a competitive manner. The high frequency of modification, a well-organized maintenance schedule including a reasonable new feature updating frequency is important, and that schedule should be well planned for mobile application maintenanence[Li et al., 2014]. Unfortunately, there is little material discussed in this area. Only Li [2013] presents three maintenance process models by manually collecting and analyzing mobile application version history data. Those maintenance models can be used as a support for scheduling update content in a new release version. Those maintenance models are described as follow.

The emergency-oriented maintenance model is showed in Figure 3.1. Small rectangles in the versions represent different maintenance types. Each version contains different maintenance contents. The arrow in the bottom is a timeline which represents the time released versions. Two rectangles in the timeline show the interval between two releases. This model is used to deal with various types of emergent issues within maintenance phase including fixing bugs and system crashes and adapting to new environment as soon as possible. In this model, the corrective maintenance is a dominant maintenance type. Moreover the frequency of different maintenance types often randomly happens. The model could be used when the users' request are very urgent.

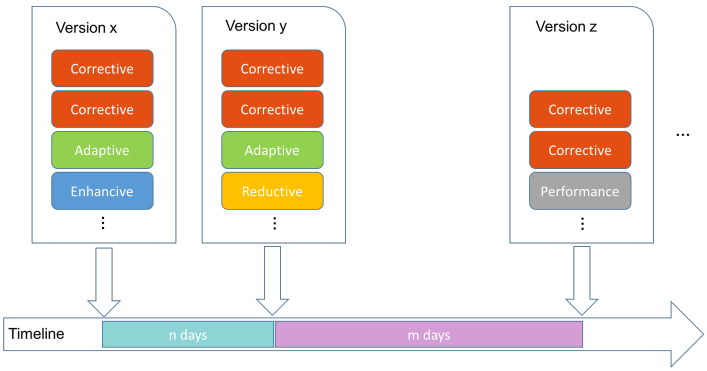

Figure 3.1. The emergency-oriented maintenance model(adapted from [Li et al., 2014])

The event-oriented maintenance model applies when more new features and functions are added release versions. Often, those features are related to holiday themes. The dominant maintenance type is the enhancive maintenance, and it only updates during some specific

day or season such as New year, Christmas, etc.. Which is more obvious in mobile games. Besides of the holidays, more customized events updates are also possible.

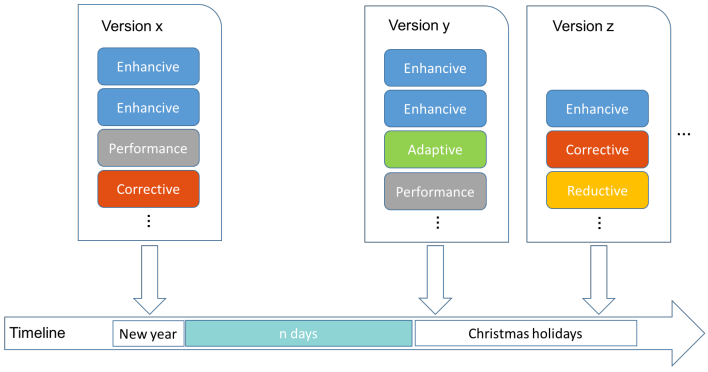

Figure 3.2. The event-oriented maintenance model(adapted from [Li et al., 2014])

Figure3.3 shows the constant maintenance model. The constant maintenance model is a regular but disciplined maintenance model that is suitable for most of the teams and projects. It has a regular release with constant time and new features. In this type, any maintenance content can be arranged in one version.

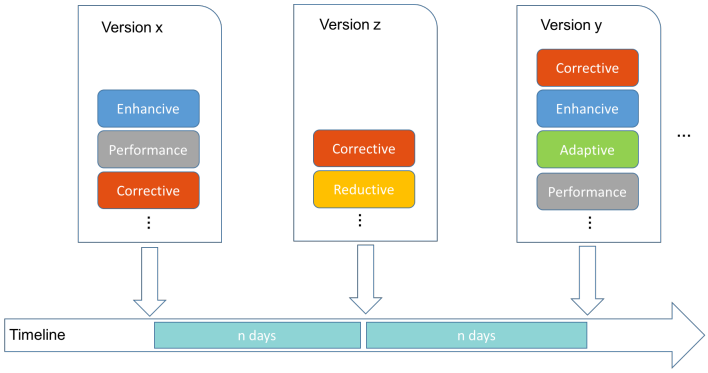

Figure 3.3. The constant maintenance model(adapted from [Li et al., 2014])

3.3 Mobile application version history data

Before introducing data mining methods which are used for exploring the data, this section explains what mobile application version history data is and how the data can be analyzed.

Mobile application version history is the data containing full records of when(the version release date) and what(the version description) an application is updated. And all of them are stored in a textual format, and recorded updates in every release since the application is launched in the marketplace. Li's research has addressed that, mobile application version history data contains valuable information in exploring mobile application maintenance practice. Figure 3.4 shows an excerpt of the version history data of a mobile game called Clash of Clans.

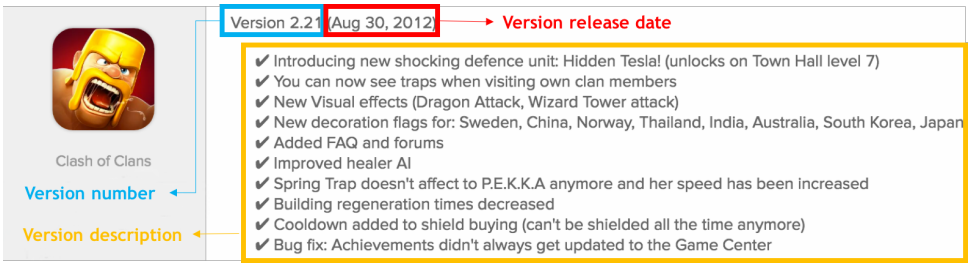

Figure 3.4. Version history of 'Clash of Clans'

As shown in Figure 3.4, each version consists of version numbers, version release dates and version descriptions. The content in the blue rectangle is the version number, content in the red rectangle is the date of updating and the content in the yellow rectangle includes version descriptions of all releases in the specific version. When an application provider releases a new version, the version number is the only identity for that version. Version release date is the time when the new version is launched on the 'App Store'. Version descriptions include the update content of that version, it implies the maintenance activities in each release version.

3.3.1 Source of the data

'App store' and 'Google play' are both official digital distribution platforms. One is for iOS users, and another is for Android users. Because 'App Store' is launched earlier than 'Google play,' the version history from 'App Store' is recorded in a longer timeline. For example, the version history of mobile game Clash of Clans, can be tracked back to June 13, 2012, while to the 'Version 7.65' released on April 30, 2015 in 'Google play'. The complete version history records are more proper in studying mobile application maintenance change through the whole application usage life. Besides, the data is only collected from the American market because of rich resources and mature methods for classifying text in English.

Moreover, the application id is collected from the website as shown in Figure 3.5. The application version history data is not directly downloaded from this website because no API for individuals is supported by iTunes. The version history data is crawled from another website called 'App Annie'[1]. After a manually comparison, we found the 'App Annie' keeps identical version history data as the original version history from 'App Store'.

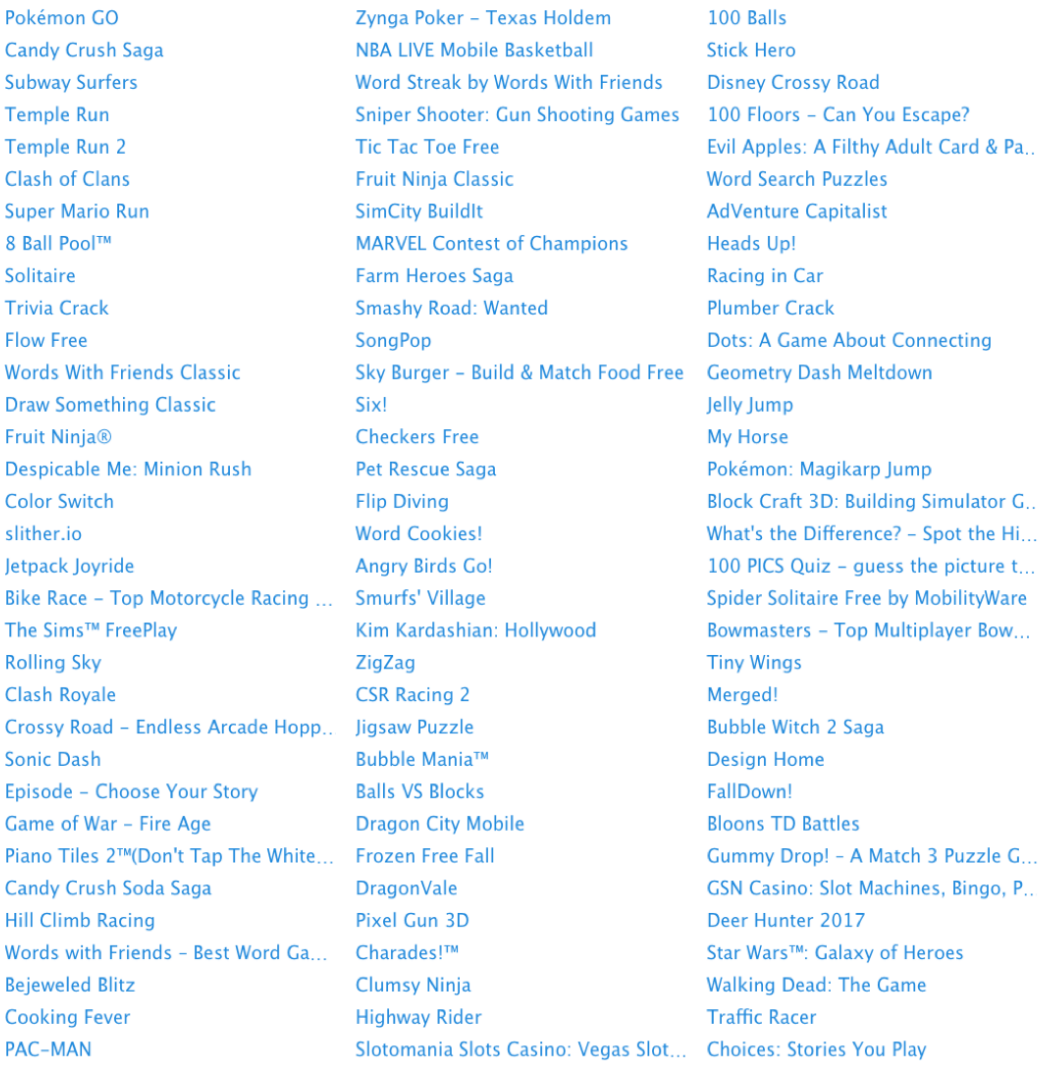

Figure 3.5. Part of 238 popular games from iTunes

Different applications may have different styles of version description. Most of them organize version descriptions in a fixed and tidy manner such as Clash of Clans, Temple Run[1], Stick Hero[2] etc., as shown in Figure 3.6. In each released version, updates are

[1] https://www.appannie.com
[1] https://itunes.apple.com/us/app/id420009108?mt=8
[2] https://itunes.apple.com/us/app/id918338898?mt=8

usually specified in a bulleted list. Sometimes, there contains two sentences in one bullet, the updates are calculated by the number of sentences. Moreover, some applications update a creative manner such as 'Candy Crush', it likes to tell a story when updates new features. For example, 'Join us in Twilight Tulips, our newest Dreamworld episode.'

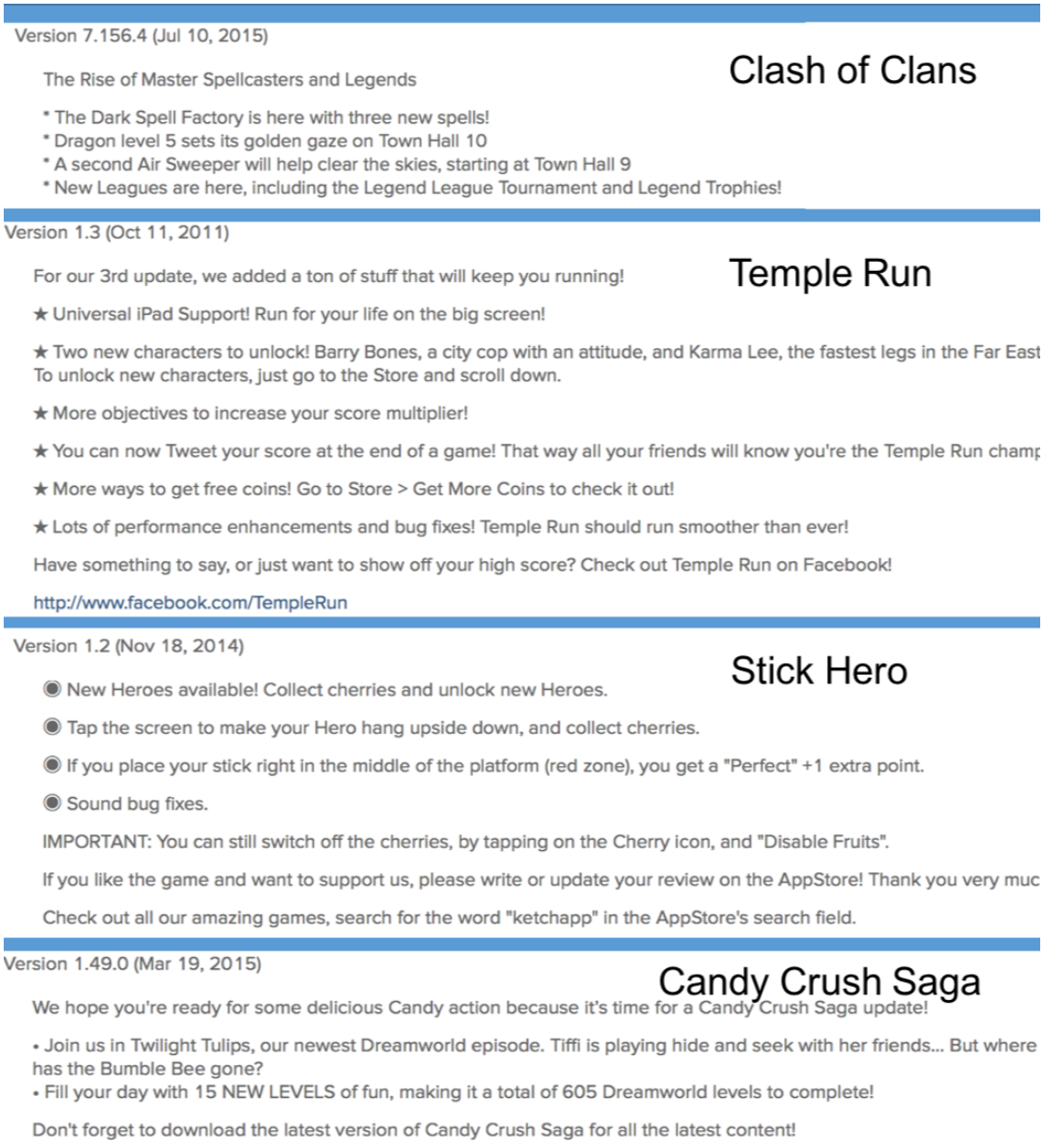

Figure 3.6. Examples of mobile application versions

Not all the updates in the version descriptions is informative. For example, The updates 'The Rise of Master Spellcasters and Legends' in Figure 3.6 is not an informative update. The common form of the uninformative updates include the advertisement of an

application, contact information of application company or some encouraging words. For example, 'If you like the game and want to support us, please write or update your review on the AppStore' is a game advertisement ,the 'http://facebook.com/TempleRun' is an contact information of game 'Temple Run' and 'Thanks you for helping Temple Run hit over 100 MILLION downloads' from the game is an encouraging words.

3.3.2 Maintenance types appear in the data

The version history data is not as official as a maintenance log, which is an internal document of application organization containing all changes of maintenance related tasks. The version history data is released aiming at helping users understand updates being done. Research from Panichella et al. [2015] indicate that users expect the information related to application features, resolved problems, helpful guidelines and enhancement changes. The version description is what users can get. Moreover, in version history data there is no update describing how a specification documentation is updated. Therefore, only maintenance types concerning the changes of the application implementation and the changes of rules can be identified in the version description. Those maintenance types are shown in Table 3.1.

Table 3.1. Software maintenance types used in this research

Business Rules	Software Properties	Documentation	Support interface
Enhancive	**Adaptive**	Updative	Evaluative
Corrective	**Performance**	Reformative	**Consultive**
Reductive	**Preventive**		Training
	Groomative		

As mentioned before, mobile application version history data illustrates efforts of application maintainers. Version description explains different updates which can uncover the type of maintenance activities and the objectives of each release version. Table 3.2 shows some version descriptions and their corresponding maintenance types.

Table 3.2. Version description updates - Maintenance type corresponding table

Version description updates	**Maintenance type**
The Dark Spell Factory is here with three new spells(Clash of Clans)	Enhancive
New heroes available...(Stick Hero)	Enhavcive
Bug fix: Achievements didn't always get updated...(Clash of Clans)	Corrective
Lots of performance enhancements and bug fixes(Temple Run)	Corrective & Performance
Optional update to remove the winter theme(Clash of Clans)	Reductive
Removed the 3th anniversary button since...(Unblock Me FREE[1])	Reductive
Universal iPad&iPhone Support(Beat the Boss 2[3])	Adaptive
Retina display support...(LEGO Star Wars[4])	Adaptive
This update includes a little polish to make the gems shine brighter and the game run smoother(Bejeweled Blitz)[5]	Performance
Improved initial load times(Deer Hunter 2014[6])	Performance
We've fixed an issue that prevented some users with older ios versions from playing enjoy(Guess The Emoji[7])	Preventive
Tailored clothing you already have no longer appears in the boutique to prevent accidental purchases(Campus Life[8])	Preventive
Don't forget to download the latest version of candy crush saga for all the newest content(Candy Crush Saga)	Consultive
Go to Store Get More Coins to check it out(Temple Run)	Consultive
Having something to say or just want to show off...(Temple Run)	Uninformative
We hope you are ready for some delicious Candy...(Candy Crush Saga)	Uninformative

[1] https://itunes.apple.com/app/id315019111?mt=8
[3] https://itunes.apple.com/app/id572534490?mt=8
[4] https://itunes.apple.com/app/id727420266?mt=8
[5] https://itunes.apple.com/app/id469960709?mt=8
[6] https://itunes.apple.com/app/id583222866?mt=8
[7] https://itunes.apple.com/app/id698848120?mt=8
[8] https://itunes.apple.com/app/id504634395?mt=8

4 DATA MINING

Currently, data mining has broadly applied in various research fields. Many software maintenance practitioners have applied data mining methods to analyzing the cost and the effectiveness of software maintenance. Shirabad et al. [2000] apply data mining techniques to learn a maintenance relevance relation among files in a software system. Mockus et al. [2003] demonstrate that system change information could be used to predict bugs. Jordanov and Jain [2010] focus on predicting maintenance time based on variables such as software platforms, service kind, complexity and service importance. All those studies achieve promising results through combing data mining and software maintenance.

Data mining is a process aiming at discovering knowledge in large data sets. Many kinds of methods or method combinations can be used for different research fields. Those methods continually grow with the disciplines it operates. For example, the natural language processing involves methods at the intersection of machine learning, statistics and linguistics[Hotho et al., 2005].

In this chapter, a general process of data mining is introduced to illustrate several significance steps in mining knowledge. Moreover, the methods used in mining mobile application version history data will be simply introduced as well.

4.1 Data mining process

Usually, the data mining process include steps of data selection, data preprocessing, data transformation, core data modeling, and result evaluation and interpretation as Figure 4.1 shows[Cheong et al., 2006]. The whole process can be rolled back at any time as the black bi-directional arrows show. Moreover, the process can be performed iteratively until the knowledge is extracted from the data.

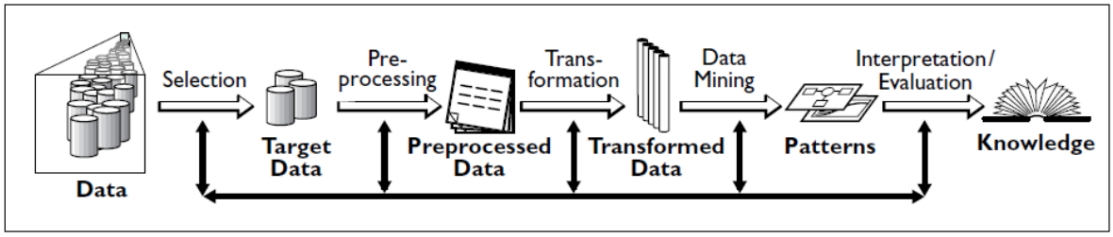

Figure 4.1. Process of data mining[Cheong et al., 2006]

In the whole data mining process, the first step is to select target data. Even though data mining deals with massive data, properly selecting target data can decrease the scale of computation and improve the analysis performance. In the data selection phase several details should be considered: selection of sites or sources, the storage format for data preprocessing phase and the legitimacy of data. After the data selection phase, the chosen data is usually called as target data or raw data.

Data preprocessing is the most time-consuming phase during the whole data mining process. It aims at transforming data into a format which can be easily and efficiently used for the succeeding phase. In this phase, plenty of details needs to be considered such as removing the wrong records, editing missing values and changing data format. For mining textual data, it is common to removing special symbols and punctuations, separating a update into a set of words. Furthermore, extracting features for a document is especially important.

Data transformation often happens together with data pre-processing phase. It often includes transforming data into forms by performing summary or aggregation operation in data transformation. For example, the daily sales data may be aggregated by month.

Core data modeling or data mining is the heart of a data mining process. Data mining problems can be separated into two categories by the requirement of training data one is supervised learning and another is unsupervised learning. Supervised learning means 'learning from a teacher', the learner receives a set of labeled examples as training data and makes predictions for all future data[Pedrycz and Chen, 2017]. The methods often used in this category are classification and regression. Unsupervised learning means the learner exclusively receives unlabeled data and intends to describe the hidden structure of the data. Clustering and dimensionality reduction are examples of this category. The output of this phase is the patterns which used to describes the global description of the data.

Pattern evaluation or interpretation is the last step, which is done for a better understanding of the analysis result. It may become an interactive process. After getting a baseline result, some parameters should be evaluated to get a better result. Data visualization methods such as points, lines or bars graphs will be used for interpreting the pattern.

4.2 Text classification

Text classification is to assign textual documents into a given set of categories such as topics or subjects. It belongs to the supervised learning category, as mentioned before, it requires a training dataset beforehand. In this research, text classification is used to assign version descriptions into proper maintenance types. The common mining methods used for text classification are Naive Bayes(NB)[Russell and Norvig, 1995], K-nearest neighbourhood(KNN)[Cover and Hart, 1967], Decision Tree(DT)[Quinlan, 1986] and Support Vector Machine(SVM)[Boser et al., 1992].

4.2.1 Classification methods

The **Naive Bayes classifier** assumes that the words appear in document d have relations with the category or class of document c. This can be described by the conditional distribution $P(d|c)$. Through the Bayesian formula, the probability of a class given the document is:

$$P(c|d) = \frac{p(d|c)P(c)}{P(d)}$$

Through this formula, the most likely probability document d belongs to c can be calculated by the value $P(c|d)$. The $P(c)$ represents the possibility of the class c occupied in all the classes. The $P(d)$ represents the possibility of the document occupied in all the documents. In order to get the maximum value of $P(c|d)$, the denominator $P(d)$ can be dropped when comparing. For example, comparing the $P(c|d)$ with ten classes, each of them is divided by $P(d)$, the $P(d)$ can act as a constant. So the maximise $P(c|d)$ is equal to maximise aggressive $P(d|c)P(c)$. Furthermore, in a document, many keywords can be chosen as features $d = (w_1, w_2...)$ for one class. With the assumption that all the classes are independent, there is

$$P(d|c) = \prod P(w_i|c)$$

For example, a update in version description includes words such as 'bug', 'fix' or 'repair' can be assigned to the corrective maintenance.[Hotho et al., 2005]

The Naive Bayes classifier is easy and simple to implement, with a set of counts. If the conditional independence assumption holds, it will quicker than discriminative models such as logistic regression. And even if not, it still often does a great job in practice. Moreover, it allows small training data set which can avoid overfitting problems. However, Using Naive Bayes is weak for learning interactions between features.

Nearest neighbour classifier does not built explicit models for different classes, it is based on calculating similarity between documents, for example calculating the common words frequency between two articles. If k similar document are considered, the method is called as k-nearest neighbour classification(KNN). The first step is to locate the k-nearest samples among the training documents by using a document-document similarity. The second step is to estimate the similarity of each class by summing the weight of the class of the k-nearest documents, as the formula shows[Kwon and Lee, 2003].

$$P(C_j|D_x) \approx \sum_{D_j \in \{k-nearst\, documents\}} sim(D_x, D_i) P(C_j|D_i)$$

Where $sim(D_x, Di)$ represents the similarity between the test document D_x and a training document D_i, and $P(C_j|D_i) \in \{0, 1\}$ represents the classification for the document D_i with the respect to category $C_j(P(C_j|D_i)) = 1\, for\, YES,\ and\ C_j(P(C_j|D_i)) = 0\, for\, NO)$

There are also several methods for measuring the similarity between documents. The simplest way is to count the number of common words between two documents. A refined version of this method is normalizing the number of common words by the total words. Another method is calculating the distance between document features included in documents.

Comparing with other classifiers, KNN classifier is robust to noisy training data., which means data has many exception values. It is also effective when the training data is small. However, it is not easy to find a suitable parametre K (number of nearest neighbors). Result can be different when choosing different k values. Moreover the computation cost of KNN is quite high because it increases by the dimension of features.[Alizadeh et al., 2009]

Decision trees are classifiers which consist of a set of rules which are applied sequentially and finally yield a decision or class. A decision tree classifier is a tree whose internal nodes are labeled by features (words occurrences in the case of text categorization), branches departing from them are labeled by the weight of the feature, and leaves are

labeled by classes. [Manne et al., 2011].

In text mining task, most of the decision tree classifiers use a binary document representation. Most of the decision tree based systems use some form of general procedure for a decision tree induction such as ID3, C4.5, and CART[Almana and Aksoy, 2014]. Firstly, the tree is built by picking a feature f. Secondly depending on the feature, the data can be divided into several subsets. Finally, invoking the pre-mentioned processes recursively until the tree node can not subdivided, and those node are labeled as leaves. The leaves are the classes to which the document belongs.

Comparing with other classifiers, the decision tree is human-understandable and robust to noisy data. However, it can easily cause the overfitting problems. Moreover, building rules is also a time-consuming process.

The support vector machine (SVM) is first introduced by Boser et al. [1992]. Currently, it has been applied successfully to text classification tasks. [Dilrukshi et al., 2013] use the SVM to classify Twitter News into 12 groups: war-terrorist-crime, economy-business, health, sports and so on with 75% accuracy. Ganu et al. [2009] apply SVM on the online user reviews and classify them into six categories: Food, Service, Price, Ambience, Anecdotes, and Miscellaneous.

The main idea of SVM is to find a linear 'hyperplane' or decision surface that can separate training data into two classes, a positive class and a negative class with the largest 'margin' between the borderline of the training data. As it shows in Figure 4.2. Two different shapes of samples are separated by a hyperplane, the margin is the distance between the samples and the hyperplane.

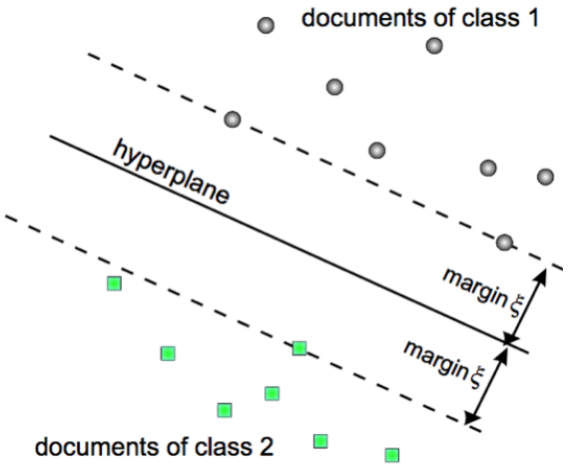

Figure 4.2. Basic concept of SVM

If such a 'hyperplane' does not exist, the data will be mapped into a higher space. A example can be shown as Figure 4.3. On the left, the samples are separated by a curve but not a line. On the right, the data is projected into a higher dimension, so a 'hyperplane' can be found for separating samples, and it can be considered as a line in one dimension.

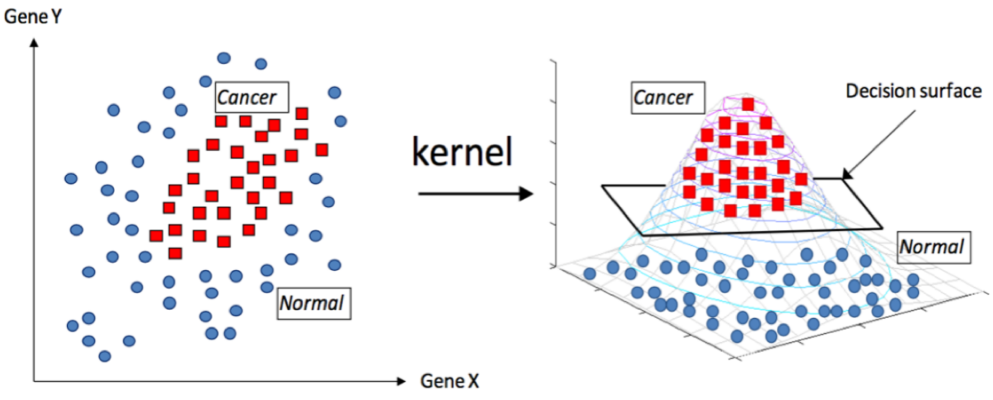

Figure 4.3. SVM with two dimensions and three dimensions

The generalized SVM classification logics shows as follow. Given a training set of instance-label pairs $((x_i, y_i), i = 1, \cdots, l$ where $x_i \in R^n)$ and $y \in \{1, -1\}^l$, the support vector machines (SVM) requires the solution of the following optimisation prob-

lem[Cortes and Vapnik, 1995]:

$$min_{w,b,\xi} \qquad \frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i$$

$$subject\ to \qquad y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

Here training vectors x_i are by the function ϕ. SVM is to find a hyperplane with the maximal margin in this higher dimensional space. $C > 0$ is the penalty parameter of the error term. Furthermore, $K(x_i, x_j) \equiv (x_i)^T\phi(x_j)$ is called the kernel function. Though new kernels are being proposed by researchers, beginners may find in SVM books the following four basic kernels[Hsu et al., 2003]:

- linear: $K(x_i, x_j) = x_i^T x_j$.
- polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$.
- radial basis function(RBF):$K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2), \gamma > 0$.
- sigmoid: $K(x_i, x_j) = tanh(\gamma x_i^T x_j + r)$.

Except for the memory limitation and interpretation difficulty, SVM often gains high accuracy with the nice theoretical guarantee. It can avoid overfitting problems. With an appropriate kernel, it can deal with data which with high dimension or with many features. Moreover, it is especially popular in text classification problems.

4.2.2 Pattern evaluation

In most of the classification tasks, there are several evaluation methods can be used as indexes to help choose suitable classifiers. A common method is to measure the performance of a classifier[Hotho et al., 2005]. It needs the predicted result dataset and the real result of the training dataset. The difference between this two result is: the predicted results are given by the machine, and the real results are given by the human. Comparing predicted results with human's results could verify the performance of the classifier. The fraction of correctly classified documents in relation to the total number of documents is called accuracy. However, using precision is not enough when the target class covers only a small percentage of the documents. If assigning each document to the alternative class, it also can get high accuracy also. Therefore, precision quantifies the fraction of retrieved documents that are in fact relevant. Recall indicates which fraction of the relevant

documents is retrieved. [Powers, 2011]

$$precision = \frac{\sharp\{relevant \bigcap retrieved\}}{\sharp retrieved}$$

$$recall = \frac{\sharp\{relevant \bigcap retrieved\}}{\sharp relevant}$$

There is a trade-off between the precision and recall. Consider two conditions as following. Most classifiers internally determine some 'degree of membership' in the target class. If only documents of high degree are assigned to the target class, the precision will be high. Because there are too much data belongs to the target class. However, many relevant documents might have been looked, which corresponds to a low recall. The F-score is a compromise of both for measuring the overall performance of classifiers. [Ahmad, 2014]

$$F = \frac{2}{1/recall + 1/precision}$$

In additional, to overcome overfitting problems, cross-validation is also a pattern evaluation method which separates data into two subsets. One subset is called training data which used to training models. Another subset is called test data which is used to test the performance of models. A more advanced approach is called K-fold cross validation. The data is divided into k subsets, and considering each subset as test data the performance can be calculated once. Then the average performance of models can be calculated[Devijver and Kittler, 1982].

4.3 Data clustering

Same with classification, data clustering is one of methods to explore patterns from the dataset. This thesis introduces them in an usage order. The goal of clustering analysis is to partition the data into clusters so that those within each cluster are more closely related to one another than assigned to different clusters. In this research, the clustering is used to grouping the maintenance type amount-time curve of different mobile games.

4.3.1 Clustering methods

K-means clustering is one of the simplest clustering methods. The procedure of K-means clustering follows an easy way to cluster data objects through a certain number

of clusters. The main idea is to define k centers, each one represents each cluster. For example, if the data objects needs to be clustered into two clusters, then the value of k will be two. Because of different center locations can result in different results, the better choice is to place them as far as possible away from each other. The next step is to take each data object belonging to a given dataset and associate it to the nearest center. When no point is left, an early cluster stage is done. At this time, K new centroids which are the barycentre of the clusters resulting from the previous step need be replaced with mean values. Now new cluster center are those means, repeating the algorithm iteratively until no more changes of mean values are done.[Rényi, 1961]

k-medoids clustering uses an actual point as the center of a cluster instead of using the mean point value as K-means. Because a mean is easily influenced by extreme values and sensitive to outliers[Jin and Han, 2011]. The basic idea is as following: Select K representative points to form initial clusters, and then repeatedly moves to better cluster representatives. All possible combinations of representative and non-representative points are analyzed, and the quality of the resulting clustering is calculated for each pair. An original representative point is replaced with the new point which causes the greatest reduction in distortion function. At each iteration, the set of best points for each cluster from the new respective medoids are generated.[Kaufman and Rousseeuw, 1987]

Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. Instead of choosing the number of k clusters, hierarchical clustering methods only require a distance method which indicates the distance of the data. As the name suggests, it produces hierarchical representations in which the clusters at each level of the hierarchy are created by merging clusters at the next lower level. At the lowest level, each cluster contains a single observation. At the highest level, there is only one cluster containing all of the data.[Hastie et al., 2009]

Hierarchical agglomerative clustering or HAC is one of widely used methods in clustering task[Rokach and Maimon, 2005]. It treats each data object as a singleton cluster and then successively merge (or agglomerate) pairs of clusters until all clusters have been merged into a single cluster that contains all observations. The logic of HAC is as follow: Firstly, it considers each observation as a cluster $C_i, i = 1, 2, ...n$. Then finding the nearest two clusters C_i, C_j use distance measurements and combining them as a new cluster $C_{i,j}$. The distance value of the new cluster can be the maximum value of C_i, C_j which is called compete-linkage clustering. The distance value of the new cluster can be the maximum value of C_i, C_j which is called single-linkage clustering. The distance value of the new cluster can be the average value of C_i, C_j which is called average-linkage clustering or

UPGMC. If the number of clusters is more than the number of clusters that we expect, iteratively repeating the formal two steps until the number of clusters decreases as we expected.

The HAC is can be well represented as a dendrogram as shown in Figure 4.4. Each merge is represented by a horizontal line. The value of the horizontal line indicates the distance between two clusters. The number of intersections of dot line and vertical line represent the number of clusters. The number of clusters can be chosen on the basis of the expected result of a data mining process or human experience. In Figure 4.4 there are four clusters, they are cluster 1 which contains points 1,4,9,10,5; Cluster 2 contains points 6, 3, 2, 11; Cluster 3 with single point 8 and cluster 4 contains point 7,12.

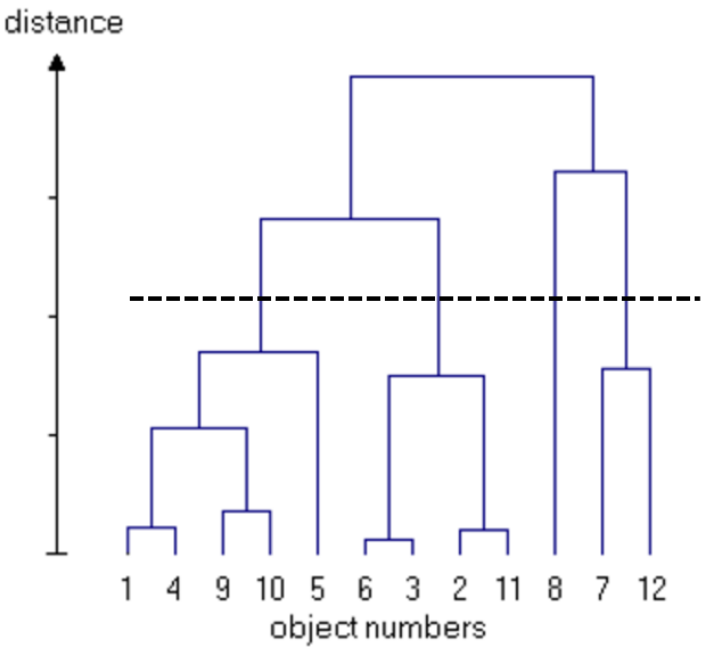

Figure 4.4. Dendorgram example

4.3.2 Distance measurements

Generally, the distance matrix is often calculated with Euclidean distance. The Euclidean distance between **p** and **q** with n dimension can be described as follow.

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Clustering maintenance type - time curve is the main goal of this research. Comparing distance with two curves, Euclidean distance is not such proper. Imagining the situations shown in Figure 4.5, the left part is aligned with Euclidean distance. The right part is aligned with dynamic time warping(DTW). It is apparently that the two curves are similar to each other, the only difference is the upper curve a slower than the bottom one. However, the Euclidean distance can not adapt this situation, because it just calculates the distance between data with the same x coordinate. DTW can solve this problem by firstly calculating distances from a single point in the bottom curve to all the points in the upper curve. Then, it got several series of nearest points. Finally, summarizing each series respectively the minimum value is the DTW distance required.

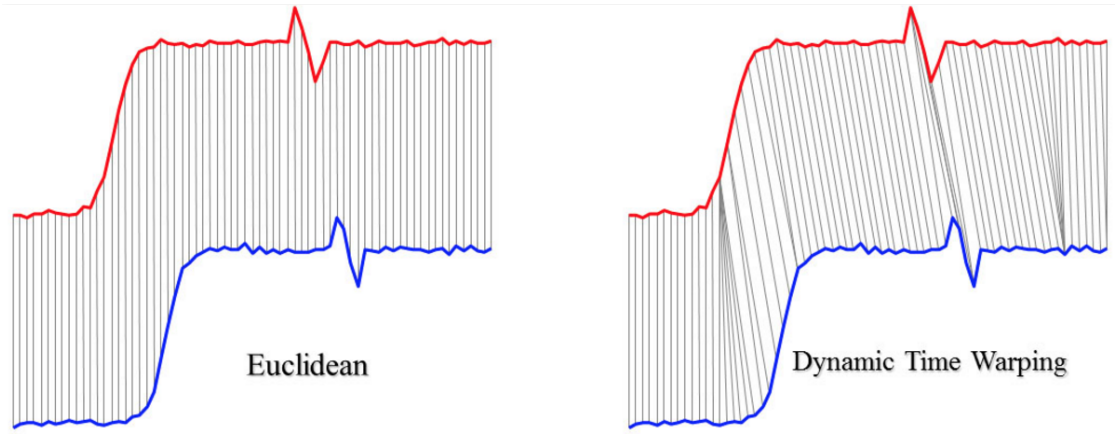

Figure 4.5. The comparison of Euclidean distance and DTW[Keogh and Pazzani, 2000]

The concrete math descriptions are shown below. Considering there are two vectors(often are two time series)**Q** and **C**, of length n and m respectively, where[Keogh and Pazzani, 2000]:

$$\mathbf{Q} = q_1, q_2, \cdots, q_i, \cdots, q_n$$
$$\mathbf{C} = c_1, c_2, \cdots, c_j, \cdots, c_m$$

To align two sequences using DTW, firstly it is necessary to construct an $n * m$ matrix where the (i^{th}, j^{th}) element of matrix contains the distance $d(q_i - c_j)$ between two points q_i and c_j. Each matrix element (i, j) corresponds to the alignment between the points q_i and c_j. This process is illustrated in Figure 4.6.

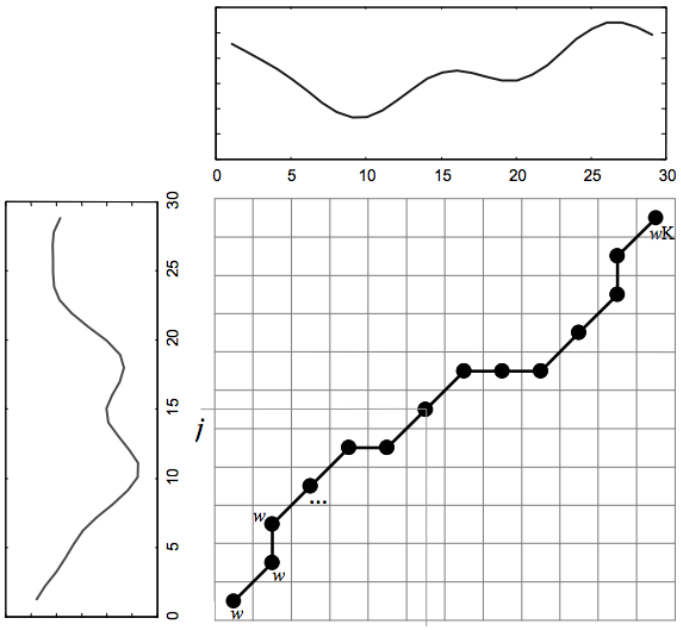

Figure 4.6. An example warping path[Keogh and Pazzani, 2000]

A warping path *W* is a contiguous set of matrix element that defines a mapping between **Q** and **C**. The k^{th} element of *W* is defined as $w_k = (i, j)_k$ so there is:

$$W = w_1, w_2, ..., w_k \; max(m, n) <= K < m + n - 1$$

There are many wraps can be calculate satisfy the conditions, however the minimized warping cost is the best result:

$$DTW(Q, C) = min\{\sqrt{\sum_{k=1}^{k} w_k}/K$$

In this research, hierarchical clustering with DTW will be used. Firstly, hierarchical clustering need not define the k value. Secondly, it is less sensitive to outliers. Moreover, the dendrogram for representing the result is clear to interpret the number of clusters and results.

5 DATA COLLECTION AND ANALYSIS

In this chapter, the data collection process and data analysis process will be described and illustrated in a working process order as shown in Figure 5.1. The first step is data collection, data is collected through a web crawler which is implemented by Python language. The second step is text classification. Text classification is to classify each update in the version description into the right maintenance type by using the maintenance types which introduced in Chapter 3. Then, counting the number of different maintenance types in each release of each application. The proportion of maintenance types can illustrate the dominant maintenance type in the mobile game application. The final step is data clustering, the aim is to understanding how maintenance activities change with time. Through the processed data from text classification stage, a maintenance curve of each application will be depicted according to the classified version update date and the amount of maintenance types. Depending on the timeline of the application since it is first launched, the data can be divided into subsets. Then clustering curves in each subset which has similar changing trends. Finally, summarizing the results from the above processes.

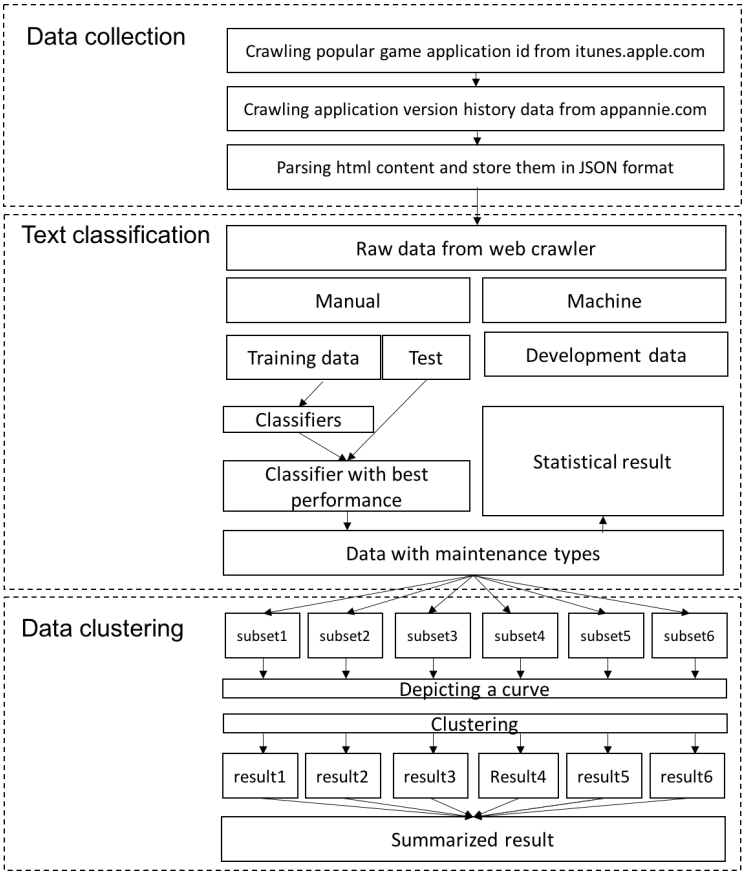

Figure 5.1. Process of text classification analysis and data clustering analysis

5.1 Data collection

Before introducing the process of data collection, a clear clarification of the data scope is necessary. In this research, the version history data from the iOS game category in America market is collected as the target data. The data collection ending date is December, 1st, 2015.

The process of data collection is shown in Figure 5.2. Because iTunes does not support any API for collecting mobile version history data, two steps are needed to collect the data. The first step is to get the id of each application from 'App Store'. Then concatenating the id with the domain name of 'App Annie', we create a new website address for the web crawler. The process of collecting and parsing application version history data from the new website are shown in Figure 5.2. The pseudocode of this process is shown in Appendix 1.

Figure 5.2. Process of data collection

The collected data is stored in a JSON file, which saves all the mobile game version history data. The file is named as 'Game.json' and shown in Figure 5.3. Each line in the file contains the whole pack of the version history from one game.

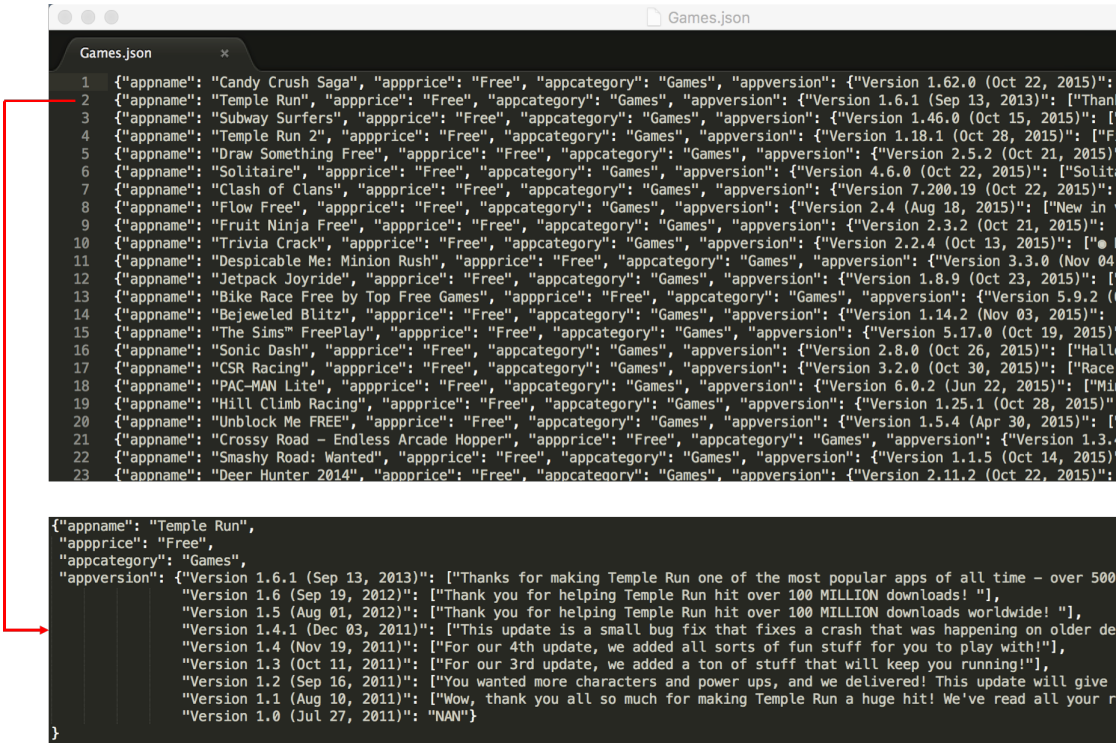

Figure 5.3. Example of textual data in JSON format

The detailed data storage structure of mobile game Temple Run is shown at the bottom of the Figure 5.3. It is a dictionary structure. The application name is the primary key and used to differentiating applications. Each application contains application price, category and 'appversion' as elements. The 'appversion' is a sub-dictionary, which contains the version release dates and version descriptions sorted in a time descending order. All the missing values are filled with 'NAN'. Furthermore, the version description is divided into sentences and stored in an array. Each sentence is consider as an update. The special situation as 'bug fixes and improve performance' will be separated into two sentence by the word 'and', therefore it is considered as two updates and has two maintenance types.

5.1.1 Overview of the data

After data collection, there should have been 240 games being collected. However, two applications are discarded automatically by the web crawler because they can not be found in the 'App Annie'. Therefore, mobile application version history data from 238 mobile popular game applications are collected. Moreover, there are many categories of games in the 'App Store' such as action game, adventure game, arcade game etc.. However there is no clear boundary between those categories. For example, 'Temple Run' is an arcade

game and also it can be found in adventure game category. Finally, data are collected from different game categories. Data collection is stopped at December, 1st 2015.

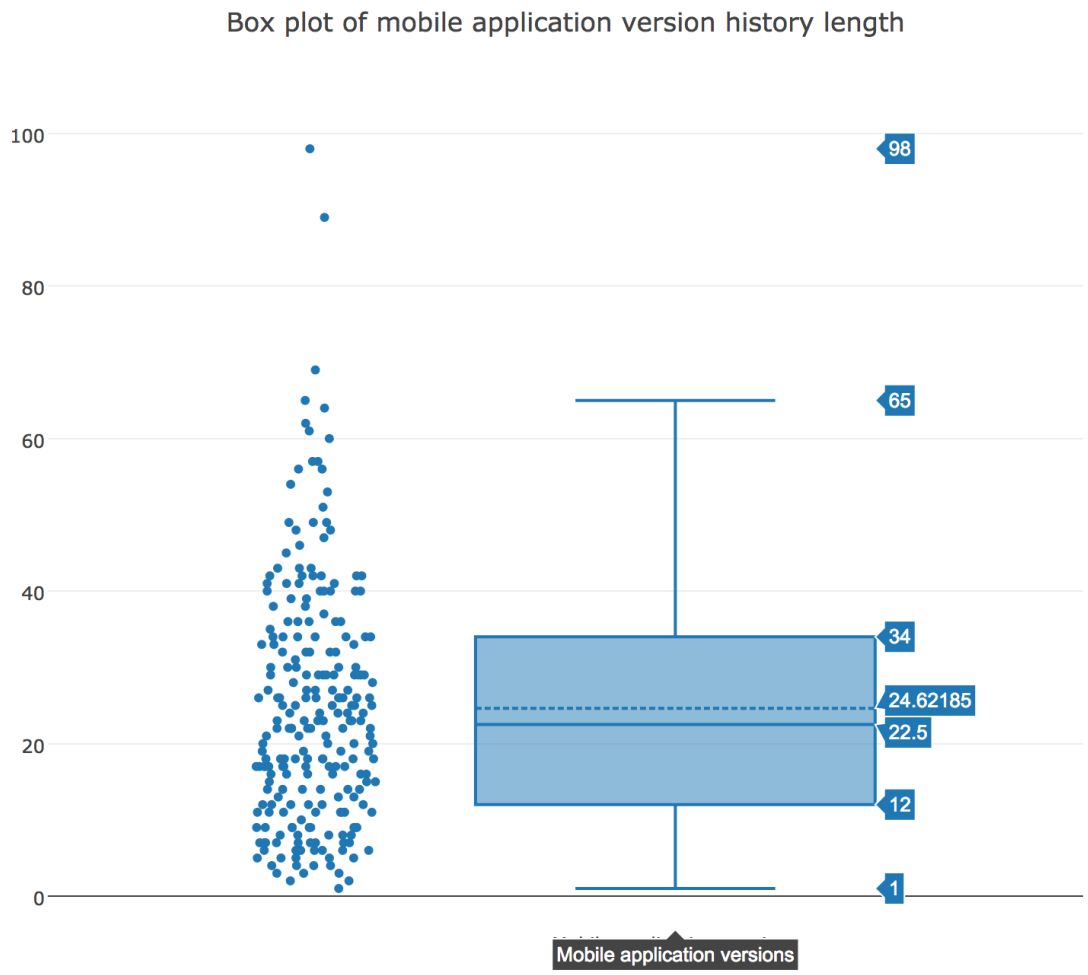

Figure 5.4. Box plot of mobile application version history data

The statistic description of the data is shown in Figure 5.4. The scatter points on the left represent the distribution of mobile application version length. If an application contains two versions, the version length will be two. Most of the applications have 12 to 34 versions. The box on the right shows the maximum, mean value, middle and minimum of version length respectively. In the dataset, there are 5860 versions in total. The application 'NBA JAM by EA SPORTS LITE' only has one version is the shortest version history. Game 'Slotomania Casino Las Vegas Free Slot Machine Games Bet Spin Win' has the longest version history with 98 versions. In average, each application has about 24.6 versions.

5.2 Text classification analysis

In the text classification phase, dealing with version description is the main concern. The text classification is to tag version descriptions into the proper maintenance types. Firstly, the raw data needs to be preprocessed. The preprocessing includes activities such as aligning all the updates with lower-case, and removing all the punctuation, numbers, ads, hyperlinks and organization contact information. Because text classification is the supervised data mining, a training data set should be built in advance. So the raw data set is separated into two approximately equal parts, with one part as the combination of the training dataset and test dataset, and the other as the development dataset. The raw data is separated horizontally, which means, for each application that has four releases, for example, the first two releases will be classified manually as training and test dataset with the rest of predicted by machine as development dataset.

After the training dataset is built, different classifiers are implemented and fed with training dataset. Based on the performance of each classifier, the one with the best performance will be chosen for predicting the result of developing data. The output of text classification is the version descriptions which are replaced with maintenance types. The detailed process, available data, and results are illustrated in following sections.

5.2.1 Building training data

In this phase, updates in the version description are assigned into eight classes as Table 5.1 shows. The first row in the table shows the classes used in text classification. Except for the uninformative class, other classes come from maintenance types in the Chapter 3. Because not all version description is useful, the uninformative class is needed to store all the uninformative data. The second row contains the abbreviations of classes which are used for naming the training data files. The third row is the corresponding number tags which are convenient for humans when tagging training data. For example number 0 represents uninforamtive training data.

Table 5.1. Classes used in text classification (1/2)

Classes	uninformative	enhancive maintenance	corrective maintenance	reductive maintenance
Abbreviation	un	en	cr	rd
Class number	0	1	2	3

Table 5.1. Classes used in text classification (2/2)

Classes	adaptive maintenance	performance	preventive maintenance	consultative maintenance
Abbreviation	ad	pf	pr	cs
Class number	4	5	6	7

The process of building training data is shown in Figure 5.5. For each application, the 'appversion' is divided to proximately equal two parts. Versions in the rectangles are chosen as training data, the rest of them is taken as the development data. We manually classify those updates into proper maintenance types, and add the numeric tag to the updates. For example, the update 'Thanks for making Temple Run one of the most popular applications of all time - over 500 million players' is uninformative for discussing maintenance types, and it is assigned to the uninformative class and tagged with number 0. The update 'this update is a small bug fix that fixes a crash that...' is a corrective maintenance and tagged with number 2. Finally, the numeric tags are appended to the updates and then stored in different text files as training data. Training data files are named with abbreviations, for example 'un.txt' is short for uninformative maintenance, 'cr.txt' is short for corrective maintenance and 'en.txt' is short for enhancive maintenance. After text classification there are 535 pieces of uninformative updates, 2699 pieces of enhancive updates, 1145 pieces of corrective updates, 43 pieces of reductive updates, 235 pieces of adaptive updates, 18 pieces of preventive updates, 304 pieces of performance updates, 118 pieces of consultive updates in the training data.

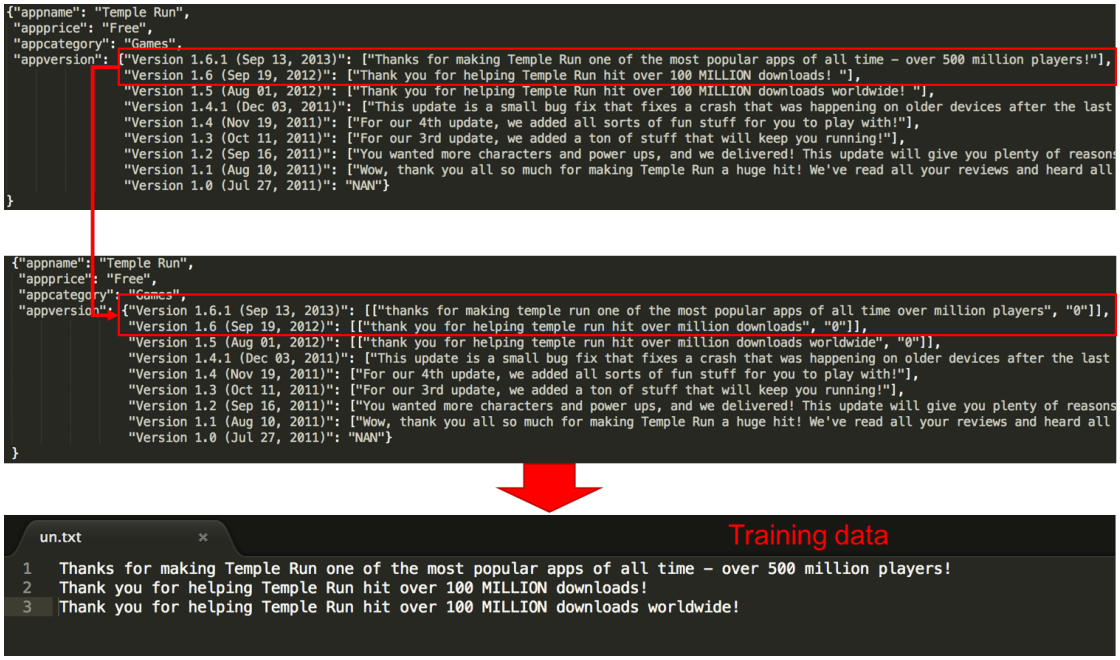

Figure 5.5. The process of building training data

When all the applications are processed, the result of training data is arranged in eight files based on maintenance type classification as shown in Figure 5.6. Each file contains updates belonging to a the responding maintenance type.

Figure 5.6. The result of training data

5.2.2 Training and comparing classifiers

After the training data is ready, the following step is to train classifiers to predict the result of the development data. In this research, four mainstream text classification methods are implemented by Python language with the scikit-learn[1] packages. Scikit-learn is a python

[1] http://scikit-learn.org/stable/

library providing different data mining methods for the scientific use, the version 0.18.1 is used in this research. There are several steps to generate a classifier by using scikit-learn. The process of training classifiers can be described in Figure5.7.

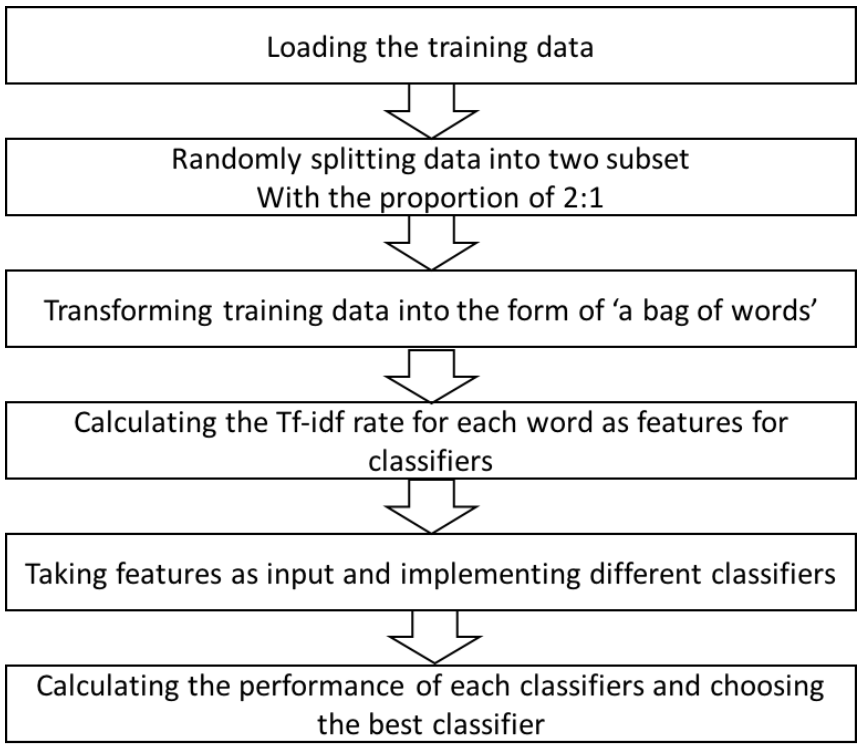

Figure 5.7. Process of training and comparing classifier

To start the training process, the training data need to be completely loaded into the text classification system. All the training data are transformed into the form of 'a bag of words' which means considering update sentences as a combination of several words. For example, a update sentence 'This is an apple' will be split as the form of 'This', 'is', 'an', 'apple'. Depending on the frequency of the word appearing in one update sentence, it is easy to predict which class the sentence belongs to. However, sometimes only considering the frequency of words is not enough, because words as 'I','a' and 'the' often appears in a high frequency, but is not helpful to find features of the training data. The tf-idf value is widely applied in text mining tasks to overcome the problem described above[Chowdhury, 2010]. Afterwards, taking those words and its tf-idf value as features, different classifiers could be generated by using the skitlearn package. Then the training data is randomly divided into three folds with an equal proportion of the data. Two of these folds are used for generating classifiers, and the rest of one fold is used for test the performance of the classifiers. Finally, calculating the performance of each classifier

and choosing a classifier with the best performance. To avoid the overfitting problem, the performance is the mean value of 3-fold cross validation. The pseudocode of this process is given in Appendix 2.

The three indexes mentioned in the Chapter 4 are used to measure the classifier performance as shown in Table 5.2. The precision means the rate of observations is classified to the right categories. The recall means the rate of the observation is not classified to the wrong categories. The F1-score is a trade-off value of precision and recall. According to the table, the SVM classifier is chosen to be the classifier for machine to learn. The machine learning process is similar to a manually learning process, the only difference is to replace human work with a SVM classifier.

Table 5.2. Performance comparison table

	Precision	Recall	F1-score
NB	0.77	0.77	0.76
SVM	**0.84**	**0.82**	**0.80**
KNN	0.79	0.43	0.41
Decision Tree	0.80	0.79	0.78

5.2.3 Available data after text classification

After finishing the predicting task, the raw data is updated. As shown in Figure 5.8, the version description objects are replaced with a specific type of array. Each element of this array represents the number of maintenance types. The elements are given in the order of enhancive maintenance(en), corrective maintenance(cr), adaptive maintenance(ad), reductive maintenance(rd), performance maintenance(pf), preventative maintenance(pr), consultative maintenance(cs), uninformative information(un), informative data and the total number of all the classes. The informative information contains all the maintenance types without useless information. The equation representation is shown below.

$$\#total = \#en + \#cr + \#ad + \#rd + \#pf + \#pr + \#cs + \#un$$

$$\#informative = \#en + \#cr + \#ad + \#rd + \#pf + \#pr + \#cs$$

$$or\ \#informative = \#total - \#un$$

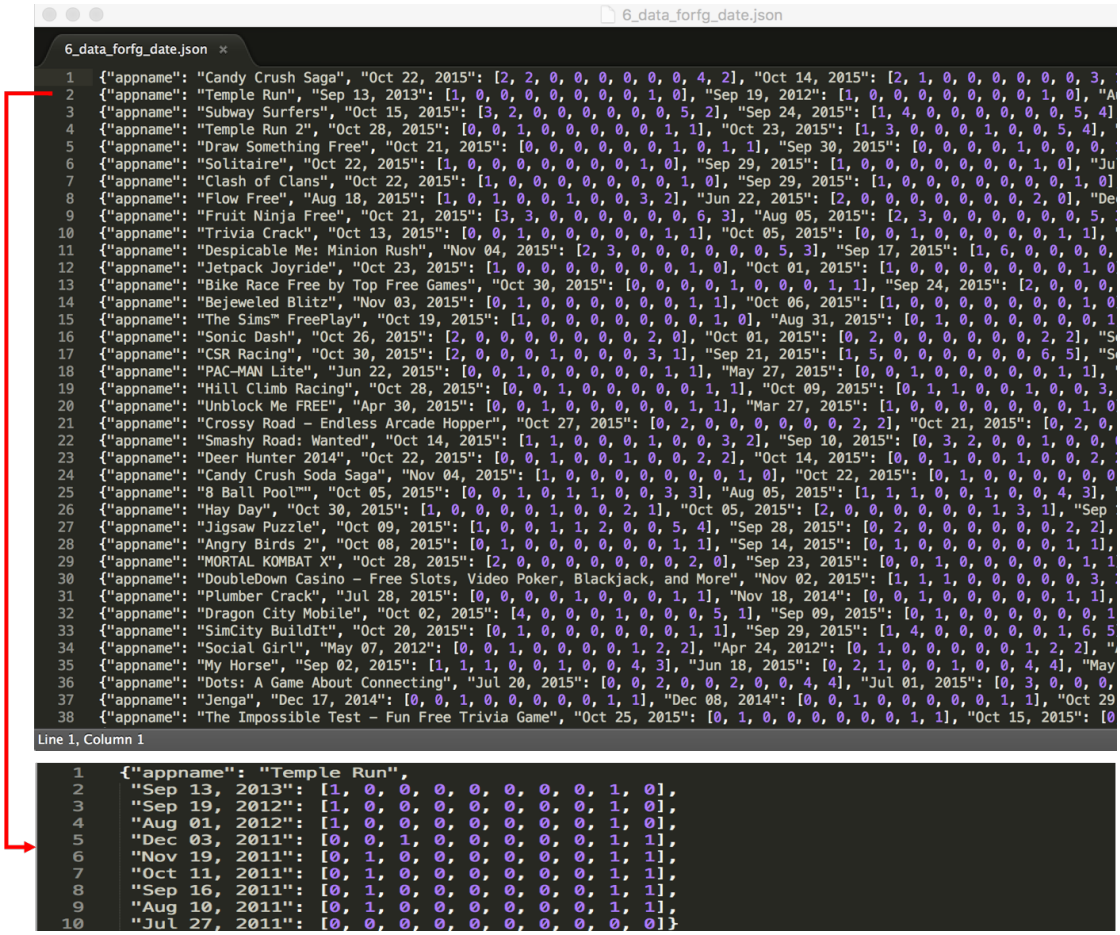

Figure 5.8. The predicted data

For a better usage, the data from this classification phase is transformed into CSV format. Every application is corresponding to a CSV file which stores all the maintenance type of every version history update of the application. An example is shown in Figure 5.9. The CSV files are the input data for the coming data clustering task. The available data for the data clustering is the 238 csv files. Each csv contains the classified maintenance types and its version release date. After the text classification, there are about 11000 updates labelled with maintenance types. And those updates will be used in the data clustering step.

PAC MAN Lite.csv

```
"date","en","cr","rd","ad","pf","pr","cs"
2015-06-22,0,1,0,0,0,0,0
2015-05-27,0,1,0,0,0,0,0
2015-05-14,4,0,0,0,0,0,0
2014-12-30,2,1,0,0,0,0,0
2014-11-03,2,1,0,0,0,0,0
2014-10-01,0,0,0,2,0,0,0
2014-09-08,2,1,0,0,0,0,0
2014-08-08,3,1,0,0,0,0,0
2014-07-10,3,1,0,0,0,0,0
2014-05-22,1,1,0,0,0,0,0
2014-03-15,0,1,0,1,0,0,0
2014-02-24,1,1,0,0,0,0,0
2013-12-21,0,1,0,0,0,0,0
2013-12-12,4,0,0,1,0,0,0
2013-10-22,0,0,0,3,0,0,0
2013-09-11,0,0,0,2,1,0,0
2013-03-26,0,0,0,1,0,0,0
2012-09-07,0,1,0,0,0,0,0
2012-08-04,0,1,0,0,0,0,0
2012-03-03,0,1,0,0,0,0,0
2011-12-07,1,1,0,0,0,0,0
2011-09-28,0,1,0,0,0,0,0
2011-06-29,1,0,0,0,0,0,0
2011-06-01,1,0,0,0,0,0,0
2011-03-16,0,1,0,0,0,0,0
2011-02-17,0,1,0,0,0,0,0
2011-02-09,1,0,0,1,0,0,0
2010-12-16,1,0,0,0,0,0,0
2010-11-30,0,1,0,0,0,0,0
```

Figure 5.9. The example of application csv

5.3 Data clustering analysis

The dataset used in the clustering task is the output of the text classification. After the text classification, version history data from 238 applications are stored in a CSV format. All the applications are stored in one file folder.

In order to investigate how the application evolves over time, an intuitive way is to calculate the similarity or the distance between maintenance curve of different applications. Before performing the task, the data are preprocessed. Afterwards, and then clustering preprocessed data into groups which have similar update trend.

5.3.1 Data preprocessing and clustering

Before clustering, data need to be processed. Firstly, some applications have few versions or are inactive during a long period of time often contain extremely short version history and can not be depicted as a curve. Therefore, the applications that fail to release a new version in half a one year until the data collection stop date, are eliminated from the clustering task. After eliminating, 190 games is used in the following step.

After calculating the age of an application based on its launch date , we divided them into six age groups and named as subset1, subset2 and so on as shown in Figure 5.10. The age

is calculated from formula shown as follow, consider x as the age and the number of the subset.

$$x = Trunc(Days\ from\ launched\ data\ to\ data\ collection\ date)\ /\ 365 + 1$$

If a game has updated for 6 years or more, it is assigned to the Subset6. There are 19 applications in the subset 1, 36 applications in the subset 2, and 48, 46, 21, 20 applications in the subsets 3, 4, 5, 6 respectively. After this step, 190 applications are depicted into curves, and 48 application are eliminated from the data set.

Date	2014.1.1	2014.4.5	2014.6.1	2014.6.20	2014.7.1	2014.7.20	2014.11.1	2014.12.1
App1	5	0	4	0	1	1	1	1

Date	2014.2.1	2014.4.2	2014.7.1	2014.7.2	2014.9.1
App2	5	4	3	2	1

Mon.	1	2	3	4	5	6	7	8	9	10	11	12
App1	5	0	0	0	0	4	2	0	0	0	1	1
App2	5	0	4	0	0	5	1	0	0	0	0	0

Figure 5.10. The example of subsets

Secondly, version update dates are accurate into the day. Applications in this step are updated at least in one year, the dimension of the data is more than 365, it is too large when calculating the DTW distance. For example, comparing two games in the subset1, more than 365 data points need to be calculated for one distance pair. To simplify the calculation, the data is aggregated by month. Doing this way can decrease the data dimension, align the release date and keep periodicity feature at the same time. There is a simple example as shown in 5.11. Two example applications, App1 and App2 are both launched at the year 2014, and they have release versions on different days. After aligning and aggregating the records, the result firstly is aligned in the same schedule, then the number of updates are summed by month.

Date	2014.1.1	2014.4.5	2014.6.1	2014.6.20	2014.7.1	2014.7.20	2014.11.1	2014.12.1
App1	5	0	4	0	1	1	1	1

Date	2014.2.1	2014.4.2	2014.7.1	2014.7.2	2014.9.1
App2	5	4	3	2	1

Mon.	1	2	3	4	5	6	7	8	9	10	11	12
App1	5	0	0	0	0	4	2	0	0	0	1	1
App2	0	5	0	4	0	0	5	0	1	0	0	0

Figure 5.11. The example of aggregate original data by month

Thirdly, all the data are scaled and depicted to the fixed size images as shown in Figure 5.12. Both Angry Birds 2 and 1010 are in a decreasing trend through scaling the data from the previous step. The coordinate of the images are omitted, because in the clustering task, the curve shape is the main concern. Finally, those curves can be compared in the same coordinate, because of the fixed image size.

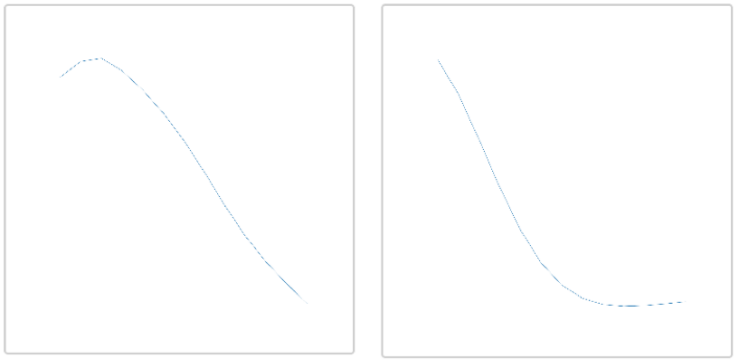

Figure 5.12. Examples of depicting data into curves

After accomplishing the three steps mentioned above, the data can be feed into the clustering algorithm. The algorithm of the clustering phase is shown in Appendix 3. Figure 5.13 illustrates the result of the subset 1 with different DTW distance measurement. The clustering is shown in a dendrogram. Each white cell is a thumbnail of the curve image and the black line under the thumbnail contains the name of the application. There are four clusters can be observed, which are C1, C2, C3 and C4. The high resolution curve image in different clusters are summarized and checked manually as Figures 5.17 - 5.14

shown. The clustering results of other subsets are given in the Appendix 4 to Appendix 8. After applying the clustering algorithm on each subset, different ages of applications have different types of update trends. The detailed results are introduced in the follow sections.

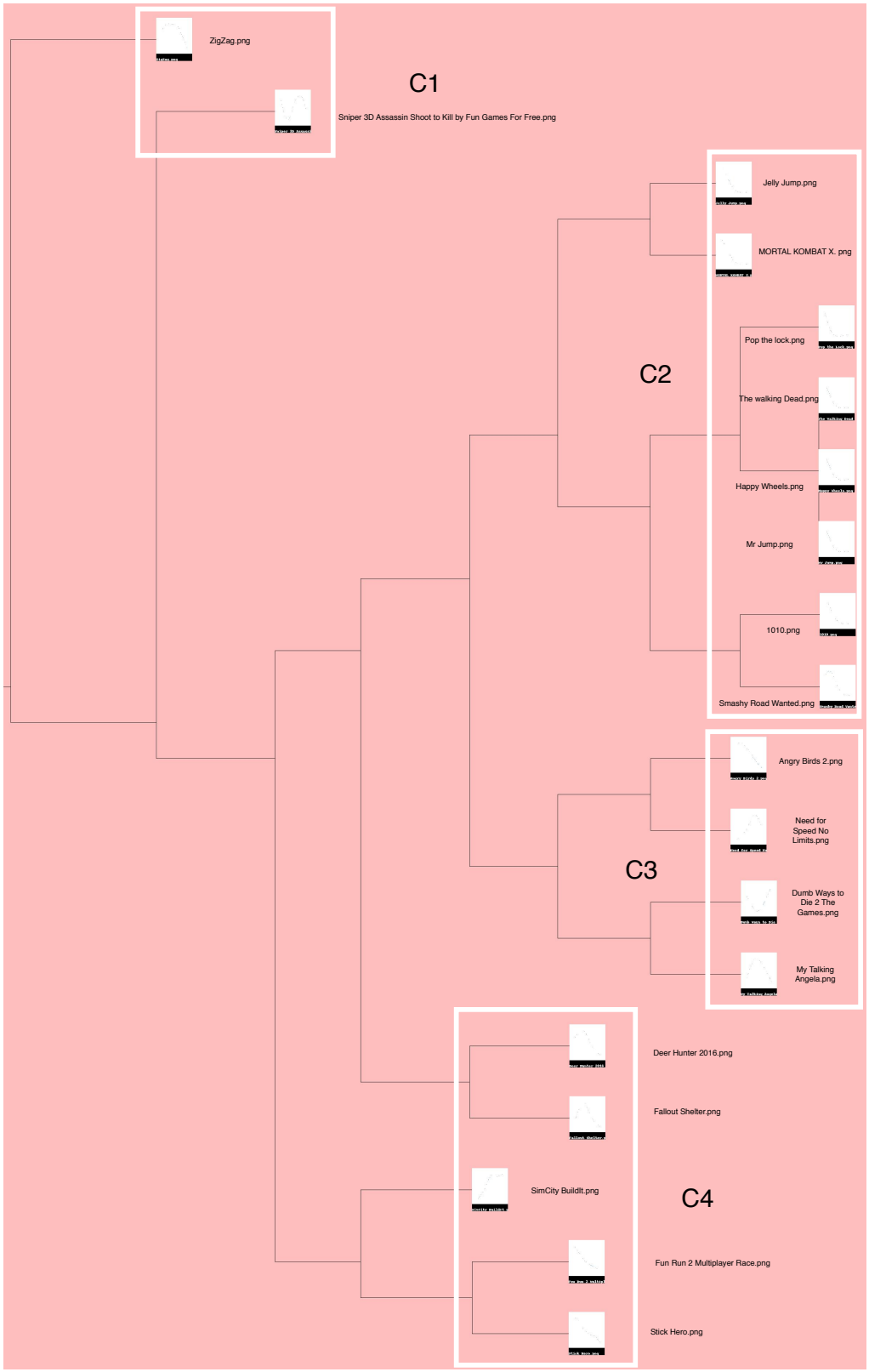

Figure 5.13. The clustering result of the subset 1

Figure 5.14. Curves in the cluster 4

Figure 5.15. Curves in the cluster 3

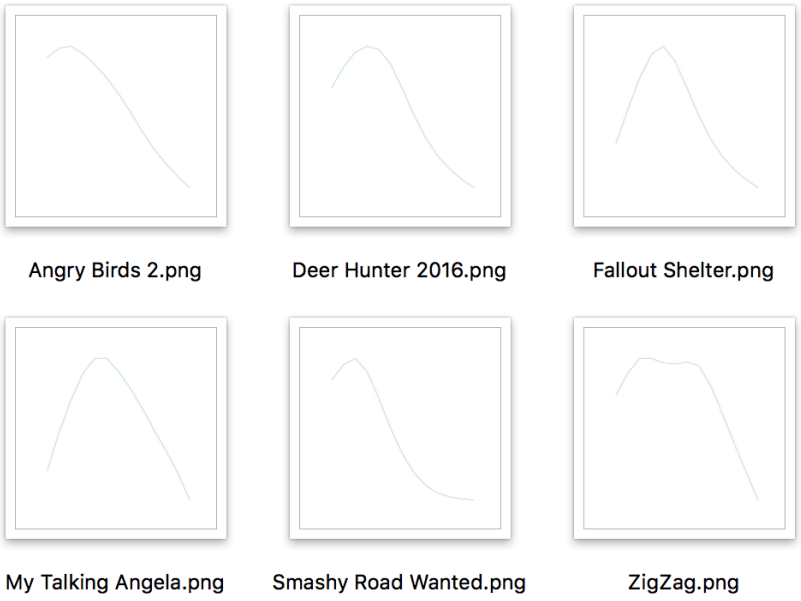

Figure 5.16. Curves in the cluster 2

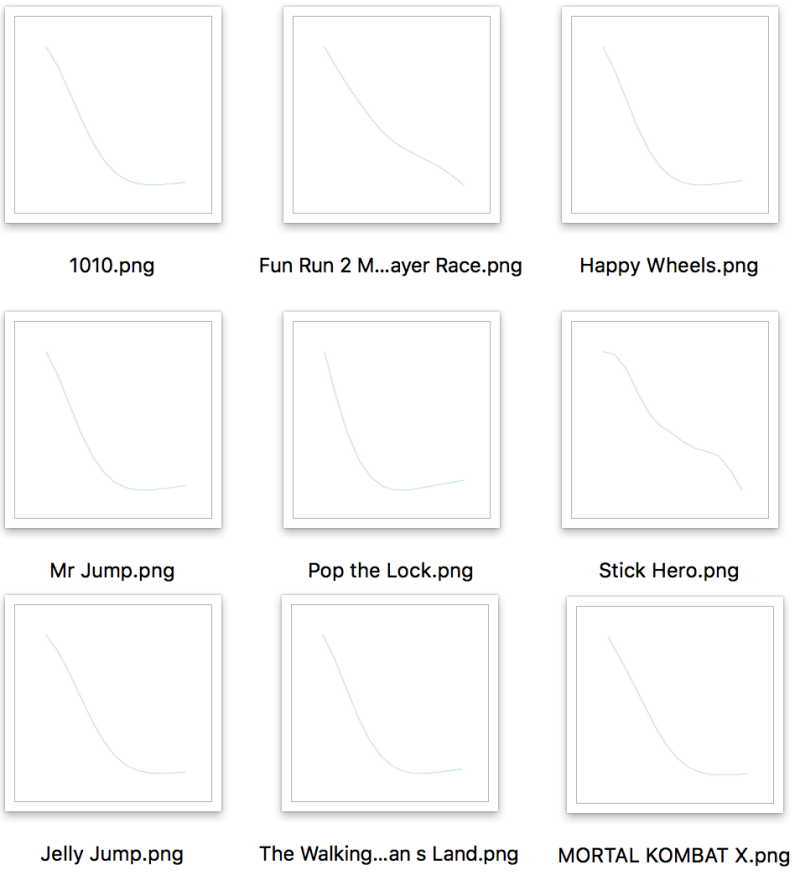

Figure 5.17. Curves in the cluster 1

6 RESULTS AND DISCUSSION

After regulating and integrating the data from the text classification phase, version history data from 238 mobile games are studied. Moreover, 12960 updates are tagged in total with 11083 pieces of informative and 1877 pieces of uninformative. Most of the version history data is informative(85.5%) as shown in Figure 6.1.

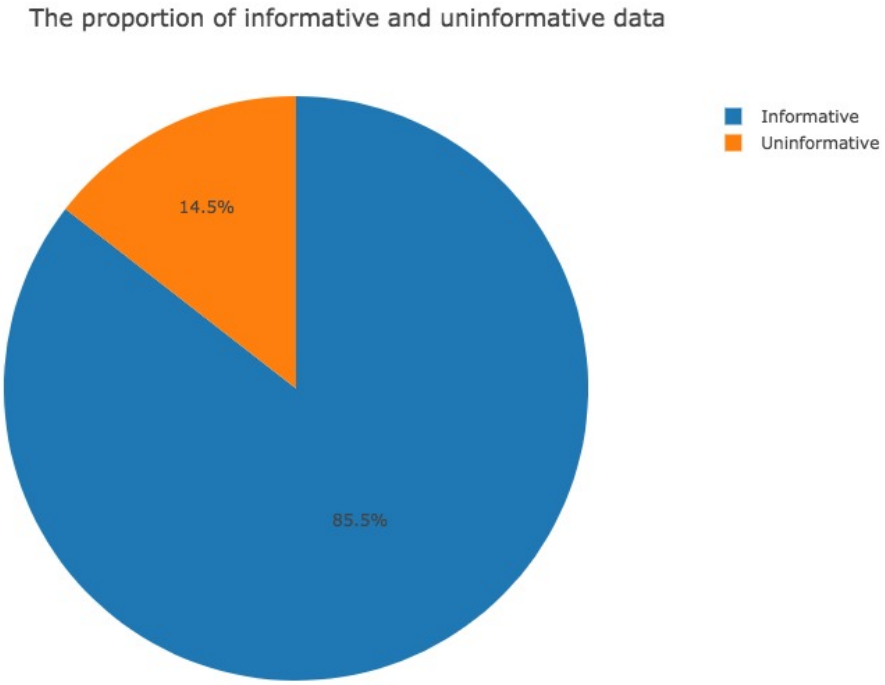

Figure 6.1. The proportion of informative and uninformative data

In the informative updates, 7152 updates are related to the enhancive maintenance, 2400 pieces of corrective maintenance and 720 pieces of performance. The Figure 6.2 describes the proportion of the different types of maintenance contained in the collected data set. Most of the maintenance works are related to enhancive maintenance. Corrective maintenance occupies 22.3% of maintenance activities. The performance is ranked third with the percentage of 6.55%. Reductive maintenance occupies 4.33% of maintenance activities.

Comparing with other kinds of utilitarian applications such as finical, health and communication, mobile games are designed for entertainment[Ahmad, 2012]. In games,

people can enjoy the process of learning game, solving problems or discovering new things[Korhonen and Koivisto, 2006]. The requests related to game stories and game rules modification appear commonly in mobile game industry such as 'Adding a new character', 'Adding new themes' and 'New tournament is beginning'. Moreover, periodically adding new characters or themes encourages a wider range of user segments to access the game and allows users pay for additional content in a flexible manner[Paavilainen et al., 2013]. Therefore, the highest proportion of enhancive maintenance illustrates that keep releasing new features is critical for mobile game providers. All the programs have bugs, so does the mobile game. It makes sense that corrective maintenance is the second frequent maintenance type. A high proportion of performance is impacted by the factor 'Time sensitive'. Activities, such as reducing some responding time, and increasing stability, are another important considerations. Besides, being impacted by the limitation of mobile application storage, mobile games also need to do a subtraction, reductive maintenance is definitely of necessity.

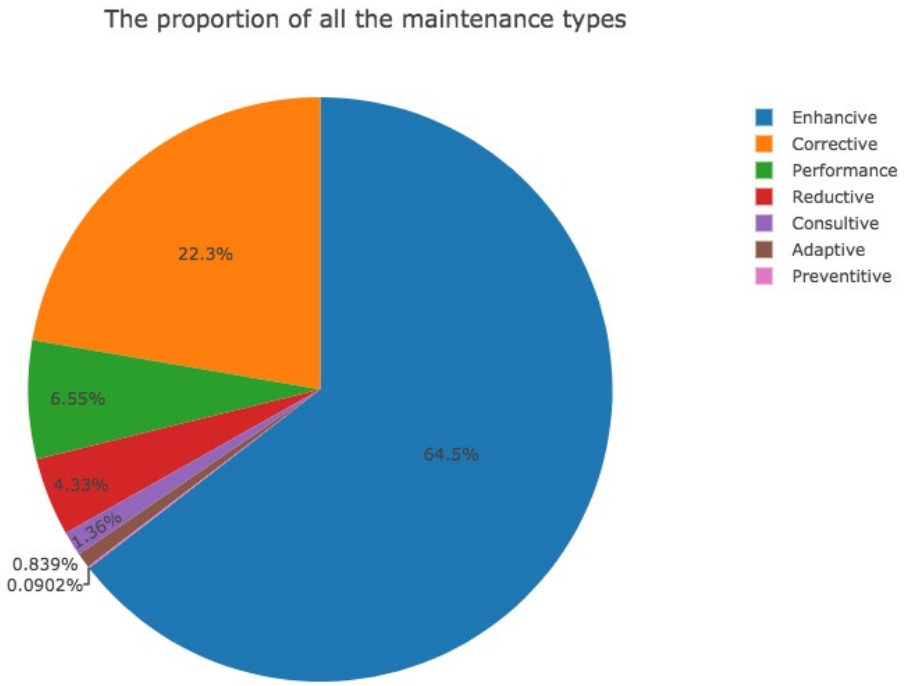

Figure 6.2. Proportion of the maintenance type from the whole data set

Going through the whole data set, Figure 6.3 shows the proportion changes of the type of maintenance in six subsets. Each year have a stack bar with the proportion of different

types of maintenance updates. For example, the subset 1 has only one stack bar, subset 6 has six stack bars respectively. The proportion of enhancive updates always keep leading. The enhancive maintenance percentage of the first year in different subsets are quite similar with the approximate proportion of 70%. Moreover, with the time changing, the proportion of enhancive maintenance decreases, the proportion of corrective maintenance and reductive maintenance increases obviously.

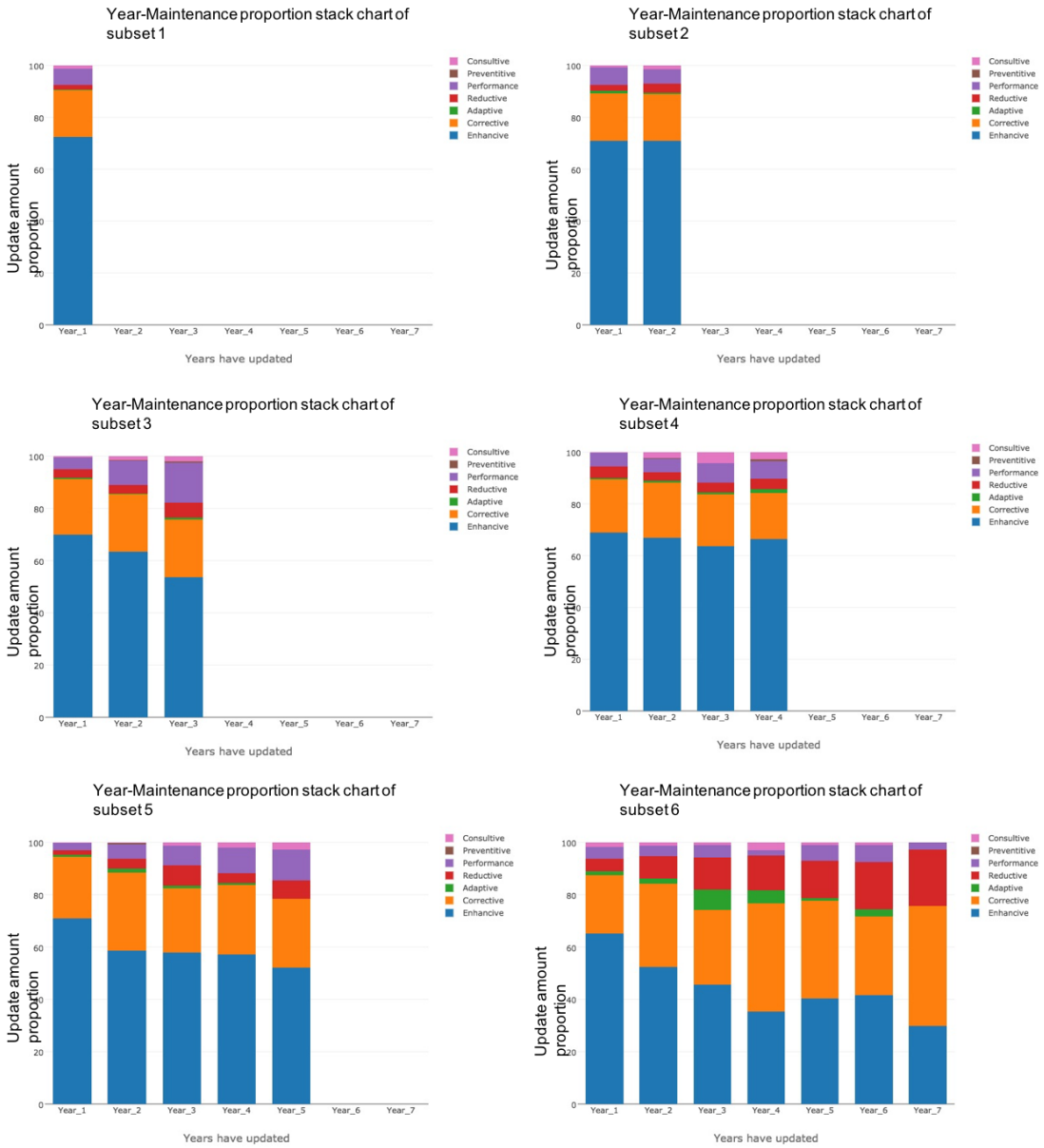

Figure 6.3. Proportion of the maintenance type from the whole data set

After the data clustering, updating trends of mobile games maintenance activities are

summarized and interpreted in each subset. There are 19 mobile games in the subset 1 in total, two of them are in trend a, another two of them update as trend b. There are six and nine mobile games update as trend c and d respectively. Results from the subset 1 indicates that there are mainly four types of maintenance updating trends as shown in Figure 6.4. Both types a and type b are in an increasing trend, while type c and d are in a decreasing trend. Most of the cases are in a declining trend such as Angry Birds 2[1], Stick Hero and 1010[2]. There are also some games update intensively such as Simcity BuitIt[3] and Dumb Ways to Die 2 The Games[4]. Because in the subset1, all the games are less than one year, the game can update in any types of trends.

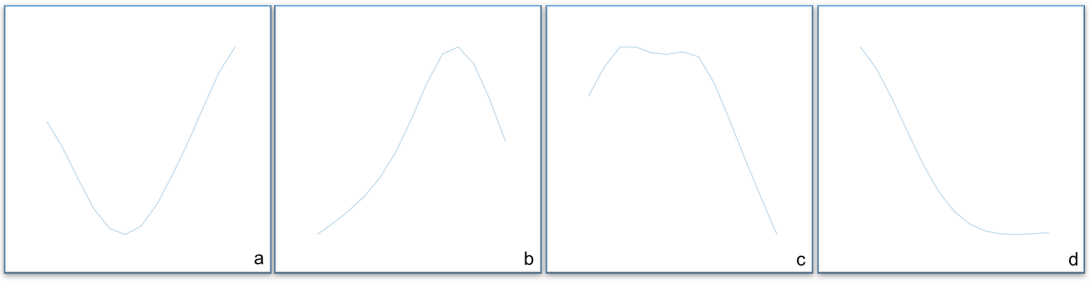

Figure 6.4. The typical updates trend of the subset 1

Comparing with the subset 1, mobile games in the subset 2, subset 3 and subset 4 experienced a longer maintenance time. The updating trends in this three subsets are similar, the difference is the distribution of games in different trend types. Therefore, they can be discussed in one group. In this group, the trend is ascending or declining with volatility as shown in 6.5. Most game maintenance activities fall in type b such as 'My Talking Tom', '4 Pics 1 Word', 'Clash of Clans' etc.. With the age of games increasing, the proportion of the type c increases and the proportion of the type a decrease as Figure 6.6 shown. Moreover, the number of peaks in different types is increasing by the age. It is worth noticing that those peaks can uncover the periodicity of the data.

[1] https://itunes.apple.com/us/app/id880047117?mt=8
[2] https://itunes.apple.com/app/us/id911793120?mt=8
[3] https://itunes.apple.com/app/us/id913292932?mt=8
[4] https://itunes.apple.com/app/us/id929346489?mt=8

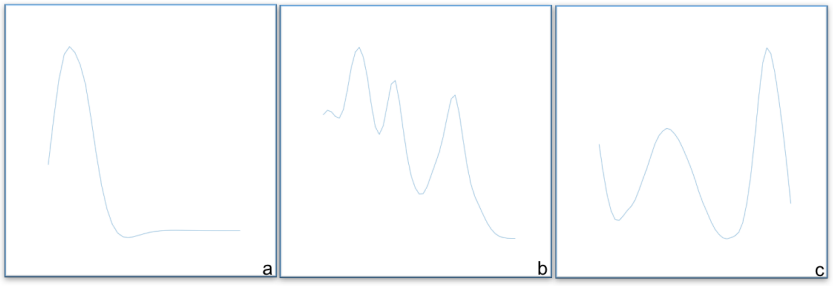

Figure 6.5. The typical update trends of subset 2, subset 3 and subset 4

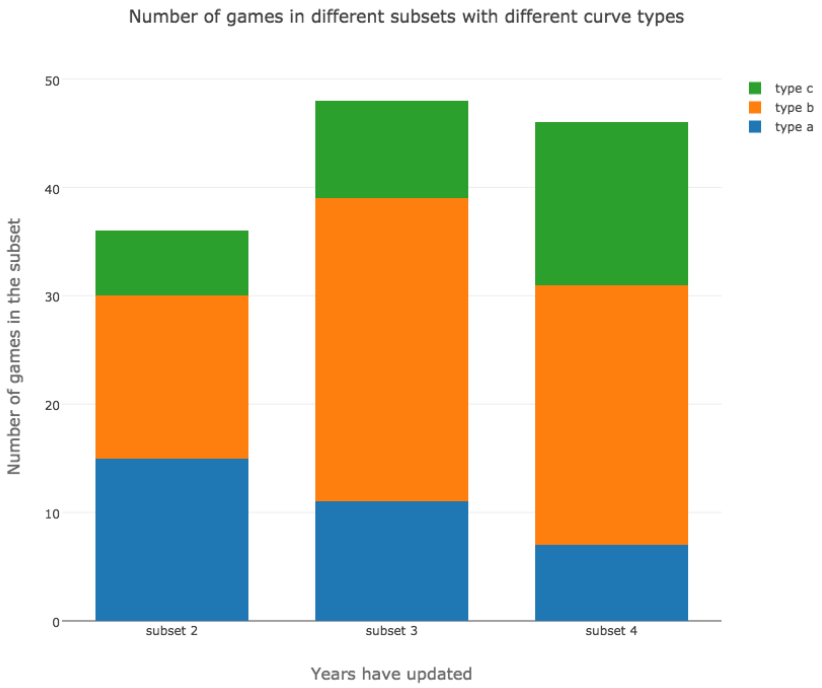

Figure 6.6. Numbers of games in different subsets with different curve types

The normal update trend type in the subset 5 are illustrated in Figure 6.7. There are 21 mobile games in this subset, and all of them could be clustered as two types. In the subset 11 mobile games update as trend a and 10 mobile games update as trend b. The type a describes games updates gradually into a stable state, the difference between peaks is changing on a small scale. According to the type b, different with the diving decline in subset 1 to 4, decreasing curves in the subset 5 are landing softly.

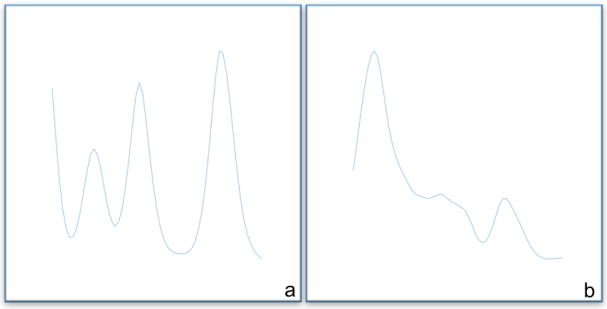

Figure 6.7. The typical update trends of the subset 5

At last, the update trend type in the subset 6 is shown in Figure 6.7. There are 20 mobile games in this subsets, 13 of them update as a decreasing trend. This trend indicates that in the long run, the energy of launching new updates decrease time by time. The games may face up the retirement decision and most of those decisions are impacted by business considerations.

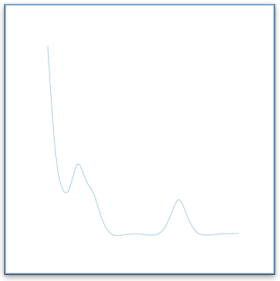

Figure 6.8. The typical update trends of the subset 6

After interpreting results from the different subsets, we combine them as a whole to get a general understanding of the mobile game maintenance. The different subsets represent applications in different ages, and can also be explained as different stages of the mobile game maintenance. The life span of mobile games comprise of four stages.

The first stage starts since the mobile game is launched. On the basis of the data from subset 1, one year or less than one year is the duration of this stage. The enhancive maintenance is the dominate maintenance type in this stage.

This stage is critical for the succeeding stages. Many factors involved in this stage need to be considered. The resources of budgetary, human and techniques can impact the

mobile game maintenance process. A big game organization may have sufficient staffs for developing and maintaining. However, developers in a small game developing team have multiple roles. Users' satisfaction is also important because mobile game suppliers could understand the role of game in the market through users' requests.

At this stage, different mobile games may have different starts as Figure 6.9 shown. Some of the games may plan many features in advance, it has more energy and acts as the first two trends. Alternatively, some of the games may be waiting for the users' opinion so that they can arrange versions in the future. Due to there are no regular trends in this stage, all the application maintenance models as emergency-oriented, event-oriented or constant raised by Li [2013] can be used.

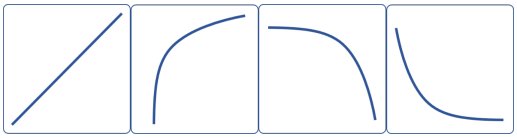

Figure 6.9. Prototype trends of the evolving stage

According to the collected data, the second stage could persist from two to four years. Mobile games in this stage grow at a fast speed. For example, a four years old game 'Big Fish Casino' releases 26 updates in 9 versions in the first year, and 46 updates in 11 versions in the second year. Moreover, more and more users join the game, requirements, bug fixing reports are overwhelming. All those modification requests are the source of updates in the future. In this stage, enhancive is the dominate maintenance type, the proportion of corrective maintenance and performance are also high.

Updates prioritization mechanism is one of the factors need to consider here. Modification requests should be prioritized based on business objectives and users' tolerance time. After the prioritization, arranging numbers of updates in one or several versions is the second. Because it is impossible to add all the modification in one version. For example, Figure 6.10 shows updates in three released versions of the game 'CSR Racing'. Updates in 'Version 1.2.0' are all enhancive maintenance, this version may have the objective of attracting users. The main objective for the 'Version 1.1.1', but it also has an objective of 'Improve system'. Finally, the objective of 'Version 1.1.0' is to improving performance.

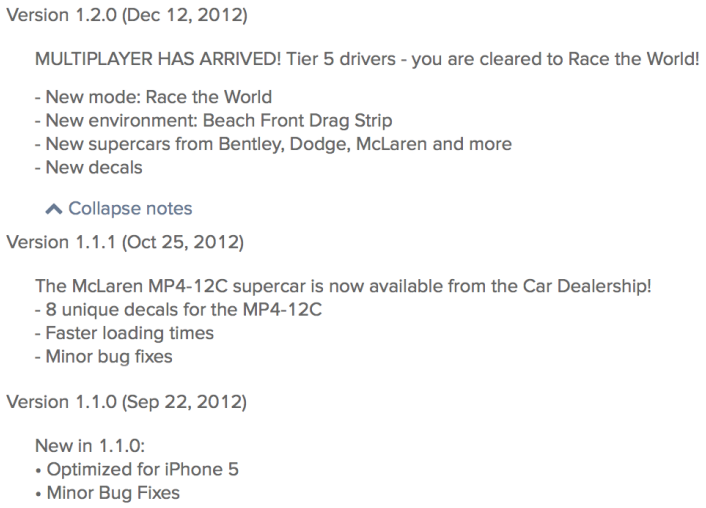

Figure 6.10. The versions of mobile game 'CSR Racing'

There are two updates trends can be summarized, as Figure 6.11 shown. Both two trends are experiencing a fast changing trend. No matter it is increasing or not, games are evolving with time. During this stage, the emergency-oriented maintenance model is proper for unexpected critical bugs or requirements users desperately need. The constant maintenance model can deal with the features requested before or a regular bug check. The event maintenance model can be aggregated in the constant maintenance model because updating in holidays is a fixed schedule for most of the games.

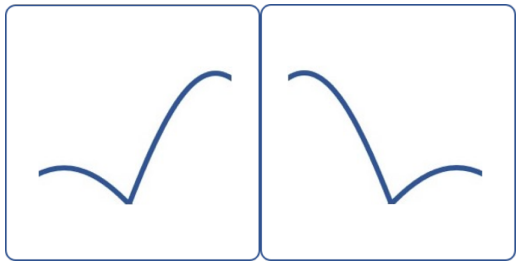

Figure 6.11. Prototype trends of the evolving stage

In the third stage, the speed and the increasing of the update amount may slow down. It is hard to find the clear boundary between the evolving stage and the mature stage. This stage is often persisted from 4 to 5 years on the basis of the data in the subset 4 and subset 5. At this stage the proportion of corrective maintenance, reductive maintenance and performance maintenance increase.

Factors impact on this stage will be the hardware and software limitations of mobile games. A game with unreasonable application size could be a problem to the user or may lead to an uninstallation. Moreover, a mobile game can not add new features endlessly. This problem can be achieved by code reconstruction. However, reconstructing code is based on the program architecture. With the time changes, the complexity of code increase, the difficulty of reconstructing code also increases. The same reason could be applied to the mobile game contents. Therefore, more corrective updates and reductive updates maintenance appear in this stage.

The trends in this stage are often of cyclical, the changing trend is stable or decays in a small scale as shown in Figure 6.12. A constant maintenance is recommended at this stage.

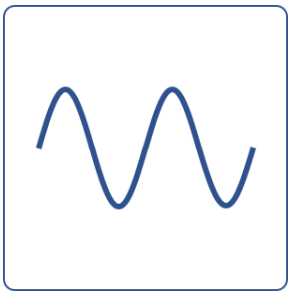

Figure 6.12. Prototype trends of the mature stage

The fourth stage is connected tightly with mobile game retirement. It is no doubt that every application will retire someday. Through a whole life span of a mobile game, the trend or the lifespan of an application is in a decline curve as Figure 6.13.

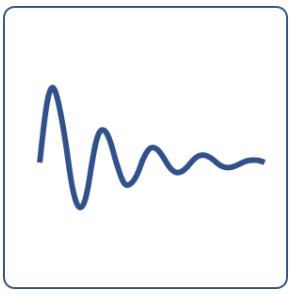

Figure 6.13. Prototype trends of the inactive stage

For mobile games or any types of applications, retirement does not mean to the application

is broken, it means they are not useful any more[Bourque and Farley, 2014]. Many factors could lead to application retirement, from the perspective of applications suppliers the business considerations is the key factor to make this decision.

7 CONCLUSION

This research is inspired by many researchers who applying data mining methods on different research fields. In this research, the mobile version history data is collected and stored in a readable and convenience format. Version history data from 238 applications are collected in total, 12960 updates are tagged. Half the updates are tagged by human efforts as a training data, which can be used in the future to solve similar problems.

Moreover, four mainstream text classification algorithms: Naive Bayes, Support Vector Machine, k-nearest neighborhood and regression tree are implemented and tested with three-fold cross validation. From the experiment results, it illustrates that the SVM classifier has the highest performance for classifying mobile application version history data.

During data clustering phase, hierarchy clustering with DTW method is used to comparing curves which have different dimensions and scales.

After applying text classification and data clustering methods on mobile application version history data, Some interesting findings are summarized. Firstly, for mobile application games, new features are always the main theme in maintenance no matter in which stage. The corrective maintenance, performance and the reductive maintenance are also not ignorable. Those types are also connected tightly with the characteristics of mobile applications. Secondly, the results illustrate applications in different stages have different focuses. Different factors impact on different stages. This information could be useful as guidance for software suppliers maintain their application products.

Indeed, this research also has some limitations. Firstly, data is separated by the age of the mobile games. There may have a better result if aligning all the data in one coordinator and separate the data into subsets as 'all the applications active in the first year', 'all the applications active in two years' and so on. It will be more clear to describe update trends in different years. Secondly, there is no enough time to studied the periodicity of the data. In the future, some time series mining methods could be applied on the version release dates.

REFERENCES

Ahmad, M. (2014). Machine learning approach to text mining: A review. *International Journal*, 4(6).

Ahmad, N. (2012). Utilitarian and hedonic values of mobile services: A preliminary analysis from the usersâĂŹ perspective. *Business & Accounting Review*, 9:69–83.

Alizadeh, H., Minaei-Bidgoli, B., and Amirgholipour, S. K. (2009). A new method for improving the performance of k nearest neighbor using clustering technique. *Journal of Convergence Information Technology*, 4(2):84–92.

Almana, A. M. and Aksoy, M. (2014). An overview of inductive learning algorithms. *International Journal of Computer Applications*, 88(4).

Basili, V., Briand, L., Condon, S., Kim, Y.-M., Melo, W. L., and Valett, J. D. (1996). Understanding and predicting the process of software maintenance release. In *Proceedings of the 18th international conference on Software engineering*, pages 464–474. IEEE Computer Society.

Bennett, K. H. and Rajlich, V. T. (2000). Software maintenance and evolution: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 73–87. ACM.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM.

Bourque, P. and Farley, R. (2014). Swebok v3. 0. *IEEE Computer Society*.

Buckley, J., Mens, T., Zenger, M., Rashid, A., and Kniesel, G. (2005). Towards a taxonomy of software change. *Journal of Software Maintenance and Evolution: Research and Practice*, 17(5):309–332.

Chapin, N., Hale, J. E., Khan, K. M., Ramil, J. F., and Tan, W.-G. (2001). Types of software evolution and software maintenance. *Journal of software maintenance and evolution: Research and Practice*, 13(1):3–30.

Chen, M. and Liu, X. (2011). Predicting popularity of online distributed applications: itunes app store case analysis. In *Proceedings of the 2011 iConference*, pages 661–663. ACM.

Chen, N., Lin, J., Hoi, S. C., Xiao, X., and Zhang, B. (2014). Ar-miner: mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on Software Engineering*, pages 767–778. ACM.

Cheong, M.-P., Sheble, G., and Berleant, D. (2006). Knowledge extraction and data mining for the competitive electricity auction market. In *Probabilistic Methods Applied to Power Systems, 2006. PMAPS 2006. International Conference on*, pages 1–8. IEEE.

Chowdhury, G. G. (2010). *Introduction to modern information retrieval*. Facet publishing.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.

Desurvire, H., Caplan, M., and Toth, J. A. (2004). Using heuristics to evaluate the playability of games. In *CHI'04 extended abstracts on Human factors in computing systems*, pages 1509–1512. ACM.

Devijver, P. A. and Kittler, J. (1982). *Pattern recognition: A statistical approach*, volume 761. Prentice-Hall London.

Dilrukshi, I., De Zoysa, K., and Caldera, A. (2013). Twitter news classification using svm. In *Computer Science & Education (ICCSE), 2013 8th International Conference on*, pages 287–291. IEEE.

Eden, A. H. and Mens, T. (2006). Measuring software flexibility. *IEE Software*, 153(3):113–126.

Ganu, G., Elhadad, N., and Marian, A. (2009). Beyond the stars: Improving rating predictions using review text content. In *WebDB*, volume 9, pages 1–6. Citeseer.

Goel, B. (2011). Integrating preventive maintenance within the product life cycle. *International Journal of Computer Applications*, 25(8):14–22.

Greer, D. and Ruhe, G. (2004). Software release planning: an evolutionary and iterative approach. *Information and software technology*, 46(4):243–253.

Grubb, P. and Takang, A. A. (2003). *Software maintenance: concepts and practice*. World Scientific.

Harjani, D.-R. and Queille, J.-P. (1992). A process model for the maintenance of large space systems software. In *Software Maintenance, 1992. Proceerdings., Conference on*, pages 127–136. IEEE.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer.

Herraiz, I., Rodriguez, D., Robles, G., and Gonzalez-Barahona, J. M. (2013). The evolution of the laws of software evolution: A discussion based on a systematic literature review. *ACM Computing Surveys (CSUR)*, 46(2):28.

Hotho, A., Nürnberger, A., and Paaß, G. (2005). A brief survey of text mining. In *Ldv Forum*, volume 20, pages 19–62.

Hsu, C.-W., Chang, C.-C., Lin, C.-J., et al. (2003). A practical guide to support vector classification.

Jeong, E. J. and Kim, D. (2007). Definitions, key characteristics, and generations of mobile games. *Encyclopedia of mobile computing and commerce*, pages 185–189.

Jin, X. and Han, J. (2011). K-medoids clustering. In *Encyclopedia of Machine Learning*, pages 564–565. Springer.

Jordanov, I. and Jain, R. (2010). *Knowledge-Based and Intelligent Information and Engineering Systems*. Springer.

Kaufman, L. and Rousseeuw, P. (1987). *Clustering by means of medoids*. North-Holland.

Keogh, E. J. and Pazzani, M. J. (2000). Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 285–289. ACM.

Kitchenham, B. A., Travassos, G. H., Von Mayrhauser, A., Niessink, F., Schneidewind, N. F., Singer, J., Takada, S., Vehvilainen, R., and Yang, H. (1999). Towards an ontology of software maintenance. *Journal of Software Maintenance*, 11(6):365–389.

Korhonen, H. and Koivisto, E. M. (2006). Playability heuristics for mobile games. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 9–16. ACM.

Kwon, O.-W. and Lee, J.-H. (2003). Text categorization based on k-nearest neighbor approach for web site classification. *Information Processing & Management*, 39(1):25–44.

Lehman, M. M. (1969). The programming process. *internal IBM report*.

Li, X. (2013). Research on agile process models in mobile application maintenance. *University of Tampere*.

Li, X., Zhang, Z., and Nummenmaa, J. (2014). Models for mobile application maintenance based on update history. In *Evaluation of Novel Approaches to Software Engineering (ENASE), 2014 International Conference on*, pages 1–6. IEEE.

Lientz, B. P., Swanson, E. B., and Tompkins, G. E. (1978). Characteristics of application software maintenance. *Communications of the ACM*, 21(6):466–471.

Mamone, S. (1994). The ieee standard for software maintenance. *ACM SIGSOFT Software Engineering Notes*, 19(1):75–76.

Manne, S., Kotha, S. K., and Fatima, D. S. S. (2011). A query based text categorization using k-nearest neighbor approach. *International Journal of Computer Applications*, 32(7):16–21.

Mockus, A., Weiss, D. M., and Zhang, P. (2003). Understanding and predicting effort in software projects. In *Proceedings of the 25th International Conference on Software Engineering*, pages 274–284. IEEE Computer Society.

Nacke, L. (2009). From playability to a hierarchical game usability model. In *Proceedings of the 2009 Conference on Future Play on@ GDC Canada*, pages 11–12. ACM.

Paavilainen, J., Hamari, J., Stenros, J., and Kinnunen, J. (2013). Social network games: PlayersâĂŹ perspectives. *Simulation & Gaming*, 44(6):794–820.

Pagano, D. and Maalej, W. (2013). User feedback in the appstore: An empirical study. In *2013 21st IEEE international requirements engineering conference (RE)*, pages 125–134. IEEE.

Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C. A., Canfora, G., and Gall, H. C. (2015). How can i improve my app? classifying user reviews for software maintenance and evolution. In *Software maintenance and evolution (ICSME), 2015 IEEE international conference on*, pages 281–290. IEEE.

Pedrycz, W. and Chen, S.-M. (2017). *Data Science and Big Data: An Environment of Computational Intelligence*. Springer.

Powers, D. M. (2011). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. Bioinfo Publications.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

Rényi, A. (1961). Proceedings of the fourth berkeley symposium on mathematical statistics and probability. *Contributions to Biology and Problems of Medicine*, 4:547–61.

Rokach, L. and Maimon, O. (2005). Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer.

Rosado da Cruz, A. M. (2016). *Modern Software Engineering Methodologies for Mobile and Cloud Environments*. IGI Global.

Russell, S. and Norvig, P. (1995). Artificial intelligence: a modern approach.

Saliu, O. and Ruhe, G. (2005). Supporting software release planning decisions for evolving systems. In *29th Annual IEEE/NASA Software Engineering Workshop*, pages 14–26. IEEE.

Salmre, I. (2005). *Writing mobile code: Essential software engineering for building mobile applications*. Addison-Wesley Professional.

Shirabad, J. S., Lethbridge, T. C., and Matwin, S. (2000). Supporting maintenance of legacy software with data mining techniques. In *Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research*, page 11. IBM Press.

Sneed, H. M. and Brossler, P. (2003). Critical success factors in software maintenance: a case study. In *Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on Software Maintenance*, pages 190–198. IEEE.

Sommerville, I. (2004). Software engineering. international computer science series. *ed: Addison Wesley*.

Stallman, R. et al. (1998). The gnu project.

Standard, I. (2006). Software engineering–software life cycle processes–maintenance. *ISO Standard*, 14764:2006.

Statista (2017). Number of available apps in the apple app store from july 2008 to january 2017. `https://www.statista.com/statistics/263795/number-of-available-apps-in-the-apple-app-store/`. 2017, June 14.

Swanson, E. B. (1976). The dimensions of maintenance. In *Proceedings of the 2nd international conference on Software engineering*, pages 492–497. IEEE Computer Society Press.

Van der Heijden, H. (2004). User acceptance of hedonic information systems. *MIS quarterly*, pages 695–704.

Wang, Q., Shen, J., Wang, X., and Mei, H. (2006). A component-based approach to online software evolution. *Journal of Software Maintenance and Evolution: Research and Practice*, 18(3):181–205.

Wilson, M. (2017). How many lines of codes is your favourite app? `https://www.fastcodesign.com/3021256/infographic-how-many-lines-of-code-is-your-favorite-app`. 2017, June 14.

Zvegintzov, N. and Parikh, G. (2005). Sixty years of software maintenance: Lessons learned. In *null*, pages 726–727. IEEE.

Appendix 1.

Algorithm 1 Collecting applications links per genre

Input: $genre\_urls, output\_filepath$

Output: $genre.txt$

```
function AppLinksGetter(genre_urls, output_filepath)
    for genre_url in genre_urls do
        tempt ←Request(genre_url)
        genre_html ←Decode(tempts)
        application_ids ← htmlParser(genre_html)
        application_links ← AppLinkTrans(applicaton_ids)
        WriteFile(application_links, output_filepath)
    end for
end function

function htmlParser(apphtml)
    app_ids ← parse application id from html
    return app_ids
end function
function AppLinkTrans(application_ids)
    app_ids ← paste id with needed domin names
    return application_links
end function
```

(continues)

Appendix 1. (continued)

Algorithm 2 Collecting mobile version history data

Input: $genre.txt, output\_filepath$

Output: $genre.json$

function APPCONTENTGETTER($genre.txt, filepath$)
  $application\_links \leftarrow$ READFILE($genre.txt, output\_filepath$)
  for $applink$ **in** $application\_links$ **do**
    $apphtml \leftarrow$HTMLTRANS($applinks$)
    DELAY$(2, 6)$
    $content \leftarrow$ HTMLPARSER($apphtml$)
    WRITEFILE($content, output\_filepath$)
  end for
end function

function HTMLTRANS($applink$)
  $request \leftarrow$ REQUEST($applink, headers$)
  $html \leftarrow$ DECODE($request$)
  return $html$
end function
function HTMLPARSER($apphtml$)
  $appname \leftarrow parse\ application\ name\ from\ html$
  $price \leftarrow parse\ application\ price\ from\ html$
  $category \leftarrow parse\ application\ category\ from\ html$
  $vnad \leftarrow parse\ application\ version\ number\ and\ date\ from\ html$
  $description \leftarrow parse\ application\ version\ description\ from\ html$
  $content \leftarrow$ DICTIONARYCONSTRACT($appname, price, category, vnad, description$)
  return $content$
end function

(continues)

Appendix 1. (continued)

Algorithm 3 Data preprocessing and built training data

for $application$ **in** $applications$ **do**
  for $appversion$ **in** $application$ **do**
    $count \leftarrow 0$
    for $sentence$ **in** $appversion$ **do**
      if $sentence! =' NAN'$ **then**
        $cleaned\_sentence \leftarrow$ CLEANDATA($sentence$)
        print $cleaned_sentence$
        $classtag \leftarrow assign\ maintenance\ types\ to\ cleaned\_sentence\ manually$
        $store\ every\ cleaned\ sentence\ in\ the\ file\ which\ named\ with\ maintenance\ tags,$
      end if
    end for
  end for
  $count \leftarrow count + 1$
  if $count \leftarrow len(appversion)/2 - 1$ **then**
    break
  end if
end for

function CLEANDATA(sentence)
  $Remove\ all\ the\ numbers\ and\ puctuations$
  $Lowcase\ all\ the\ words$
  return $cleaned\_sentence$
end function

(continues)

Appendix 2. (continued)

Algorithm 4 Training classifiers and compare classifiers

$alldata, alltarget \leftarrow$ LOADDATA($alldataset$)
$X\_train, X\_test, y\_train, y\_test \leftarrow$ TRAINTESTSPLIT($alldata, alltarget$)
$features \leftarrow$ VECTORIZER(X_train, y_train)
$classfiers \leftarrow$ GENERATECLASSIFERS($features$)
for do$classfier$**in**$classifers$
    $performance \leftarrow$ PERFORMANCETEST(X_test, u_test)
end for
$Choose\ classfiers\ with\ highest\ F1-score$
$maintienanc\_type \leftarrow$ PREDICT($development\_data$)

function LOADDATA($alldataset$)
    $Read\ data\ as\ (sentence, target\_class)\ format\ from\ all\ the\ txt\ file.$
    $alldata \leftarrow sentence$
    $alltarget \leftarrow target\_class$
    return $alldata, alltarget$
end function
function TRAINTESTSPLIT($alldata, alltarget$)
    $Randomly\ seperate\ data\ train\ data\ and\ test\ data$
    return $X\_train, X\_test, y\_train, y\_test$
end function
function VECTORZIER(X_train, y_train)
    $Spliting\ sentence\ into\ bag-of-words\ format$
    $Conting\ words\ appear\ frequency$
    $Counting\ Tf-Idf\ value\ for\ every\ sentence$
    return $features$ ▷ sentence with Tf-idf value
end function
function GENERATECLASSFIERS($features$)
    $Tryingdifferentclassfierswithfeatures$
    return $classifiers$
end function
function PERFORMANCETEST(X_test, u_test)
    $Calculatingaverage\ accuracy, recall, F1-scoreusing3foldcrossvalidation$
    return $F1-score$
end function

(continues)

Appendix 2. (continued)

Algorithm 5 Training classifiers and compare classifiers

$alldata, alltarget \leftarrow$ LOADDATA($alldataset$)
$X\_train, X\_test, y\_train, y\_test \leftarrow$ TRAINTESTSPLIT($alldata, alltarget$)
$features \leftarrow$ VECTORIZER(X_train, y_train)
$classfiers \leftarrow$ GENERATECLASSIFERS($features$)
for do $classfier$ **in** $classifers$
    $performance \leftarrow$ PERFORMANCETEST(X_test, u_test)
end for
$Choose\ classfiers\ with\ highest\ F1-score$
$maintienanc\_type \leftarrow$ PREDICT($development\_data$)

function LOADDATA($alldataset$)
    $Read\ data\ as\ (sentence, target\_class)\ format\ from\ all\ the\ txt\ file.$
    $alldata \leftarrow sentence$
    $alltarget \leftarrow target\_class$
    return $alldata, alltarget$
end function
function TRAINTESTSPLIT($alldata, alltarget$)
    $Randomly\ seperate\ data\ train\ data\ and\ test\ data$
    return $X\_train, X\_test, y\_train, y\_test$
end function
function VECTORZIER(X_train, y_train)
    $Spliting\ sentence\ into\ bag-of-words\ format$
    $Conting\ words\ appear\ frequency$
    $Counting\ Tf-Idf\ value\ for\ every\ sentence$
    return $features$ ▷ sentence with Tf-idf value
end function
function GENERATECLASSFIERS($features$)
    $Tryingdifferentclassfierswithfeatures$
    return $classifiers$
end function
function PERFORMANCETEST(X_test, u_test)
    $Calculatingaverage\ accuracy, recall, F1-scoreusing3foldcrossvalidation$
    return $F1-score$
end function

(continues)

Appendix 3. (continued)

Algorithm 6 Data preprocessing and clustering

```
data ← READ(filename)
subsets ← GENERATECLASSFIERS(data)
for do subset in subsets
    distMatrix ← DIST(subset, method = "DTW")
    hc ← HCLUST(distMatrix, method = "ward.D2")
    data ← WRITE(hc)
end for

function GENERATECLASSFIERS(data)
    for do app in data
        Duration ← CollectedDate − app[firstLauchDate]
        inactiveDuration ← CollectedDate − app[LastUpdateDate]
        if nactiveDuration > halfYear then
            del(app)
        else
            subsetnum ← days(Duration)%365
            appByMonth ← aggregate(app)
            scaleApp ← scale(app)
            appPic ← draw(scaleApp)
            subset(subsetnum).append(appPic)
        end if
    end for
end function
```

(continues)

Appendix 4. (continued)

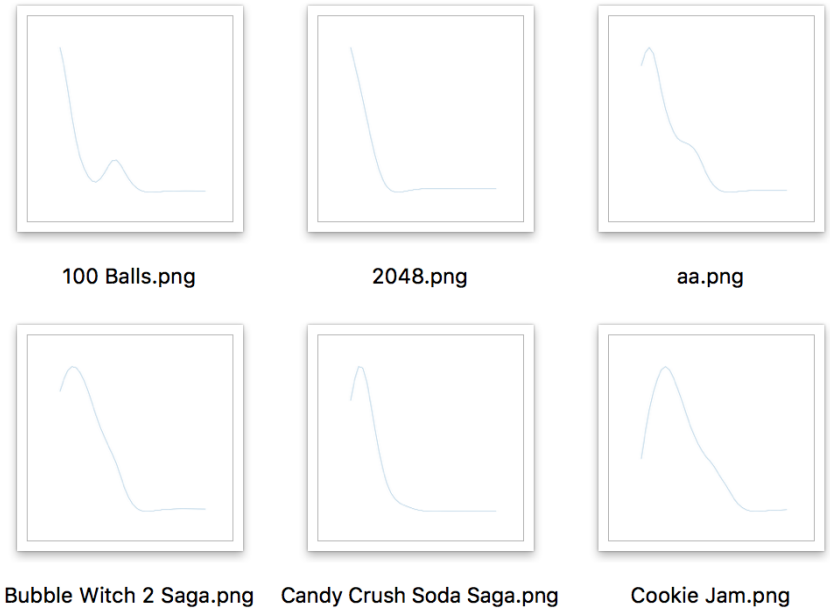

Figure A4.1. Curves in the subset 2 cluster 1

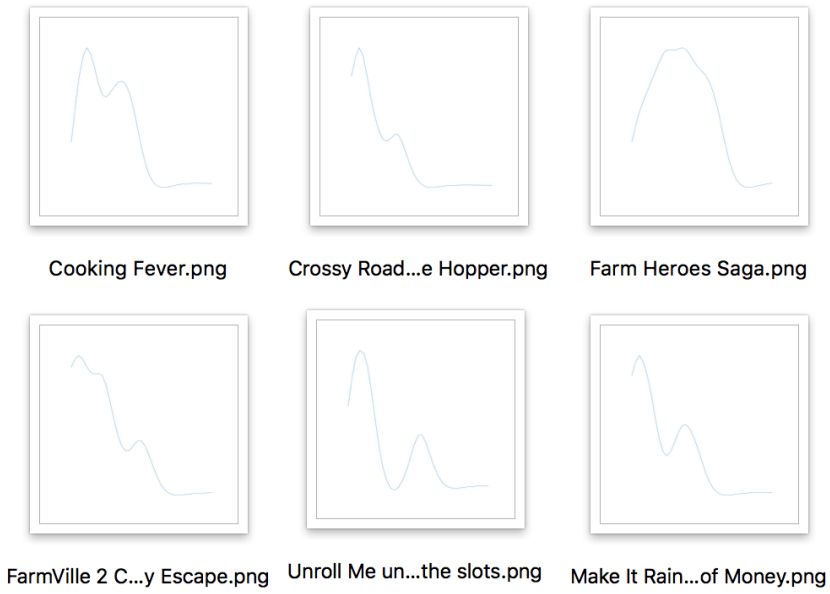

Figure A4.2. Curves in the subset 2 cluster 2

(continues)

Appendix 4. (continued)

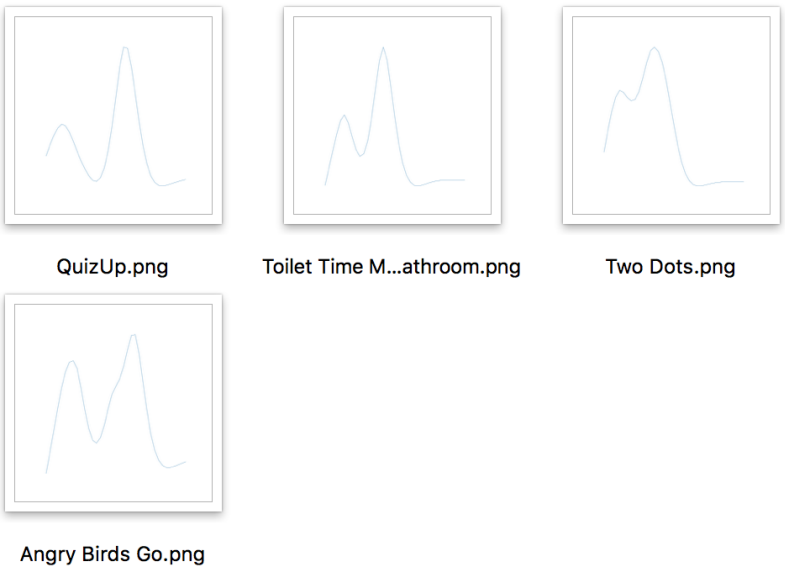

Figure A4.3. Curves in the subset 2 cluster 3

(continues)

Appendix 5. (continued)

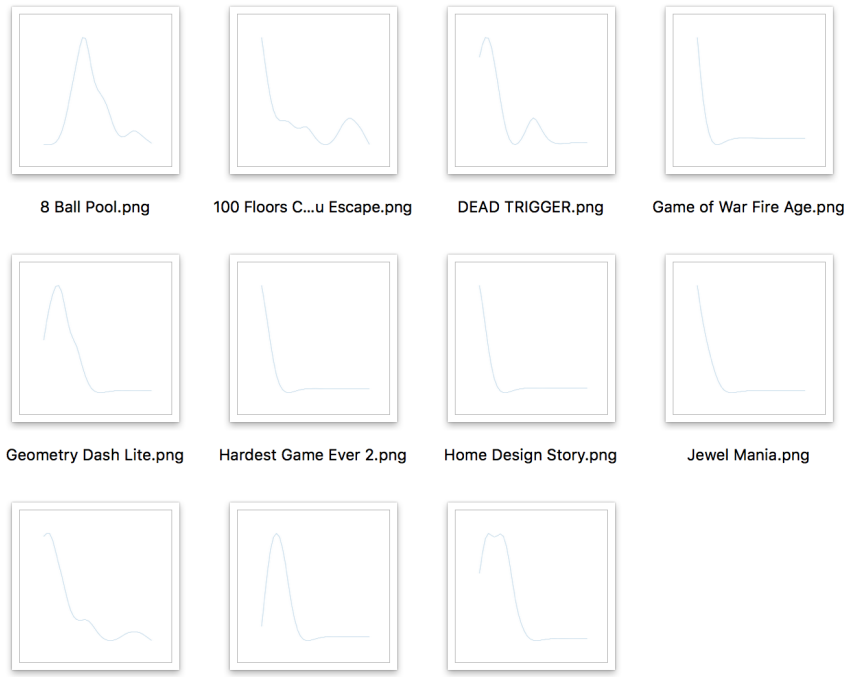

Figure A5.1. Curves in the subset 3 cluster 1

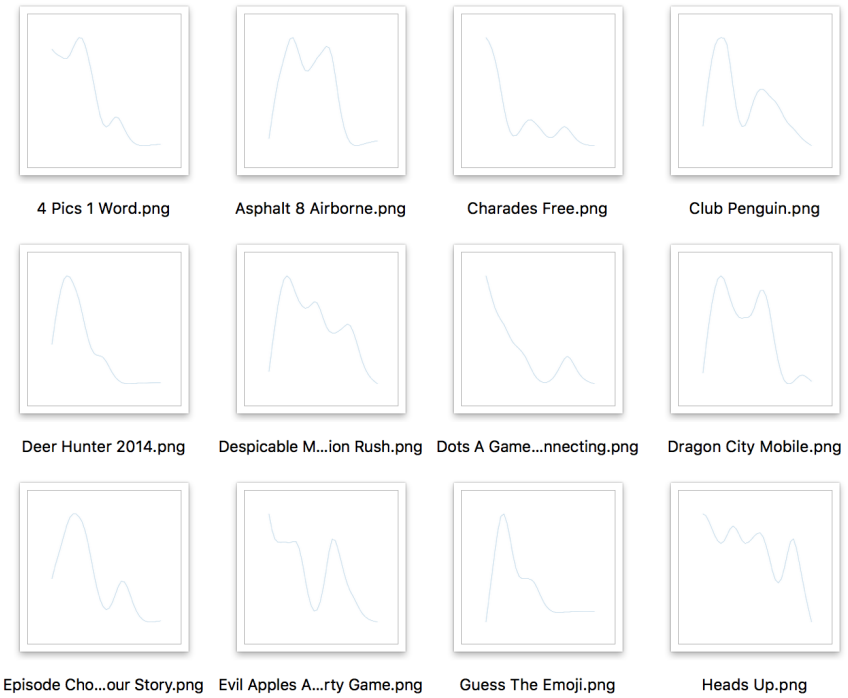

Figure A5.2. Curves in the subset 3 cluster 2

(continues)

Appendix 5. (continued)

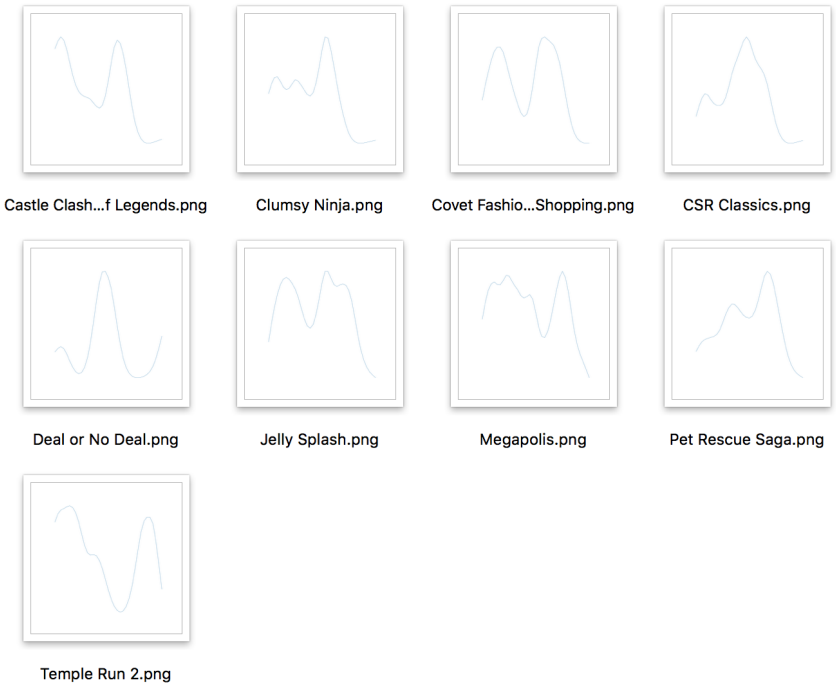

Figure A5.3. Curves in the subset 3 cluster 3

(continues)

Appendix 6. (continued)

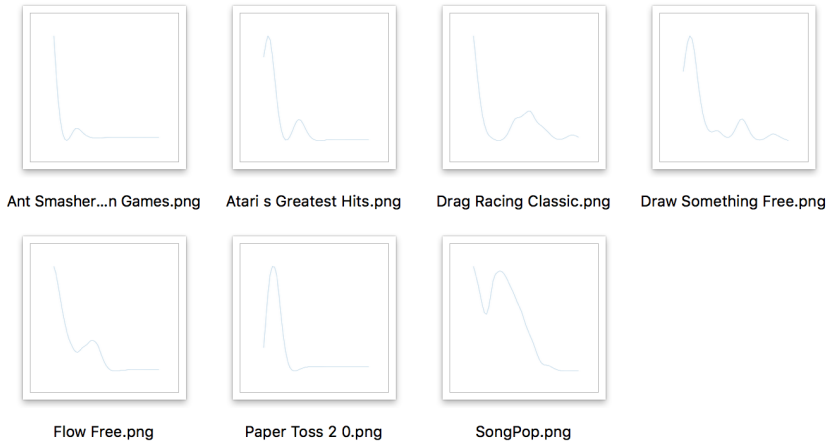

Figure A6.1. Curves in the subset 4 cluster 1

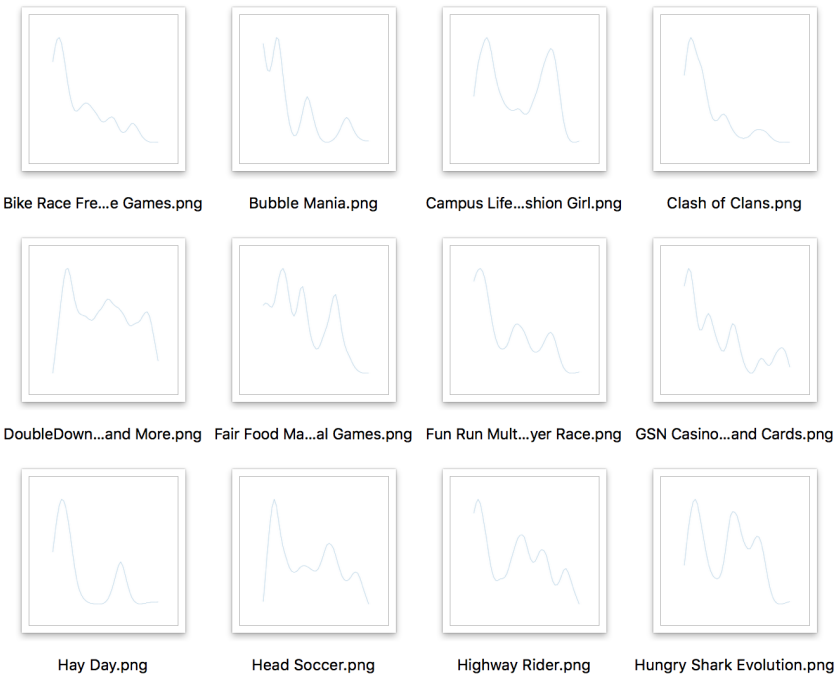

Figure A6.2. Curves in the subset 4 cluster 2

(continues)

Appendix 6. (continued)

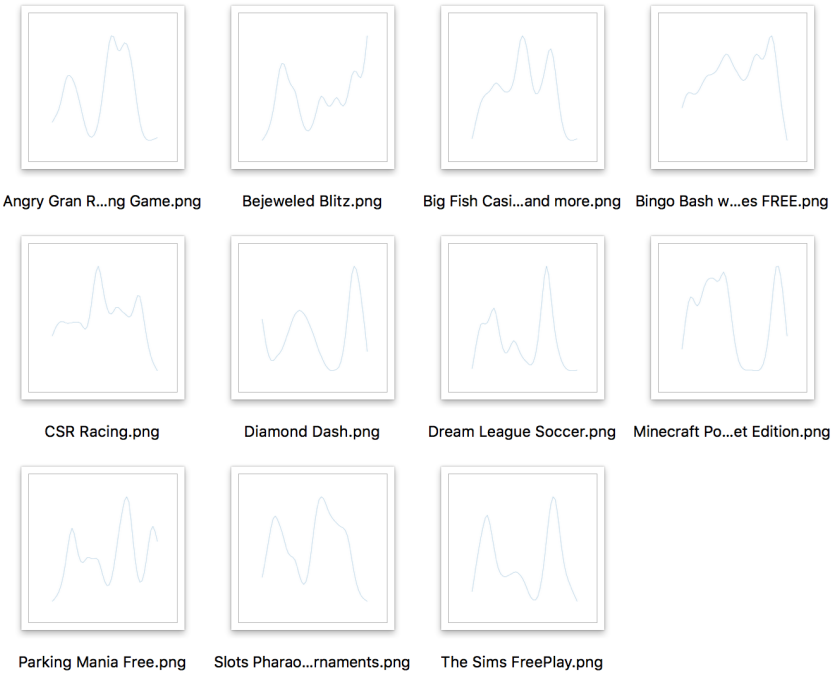

Figure A6.3. Curves in the subset 4 cluster 3

(continues)

Appendix 7. (continued)

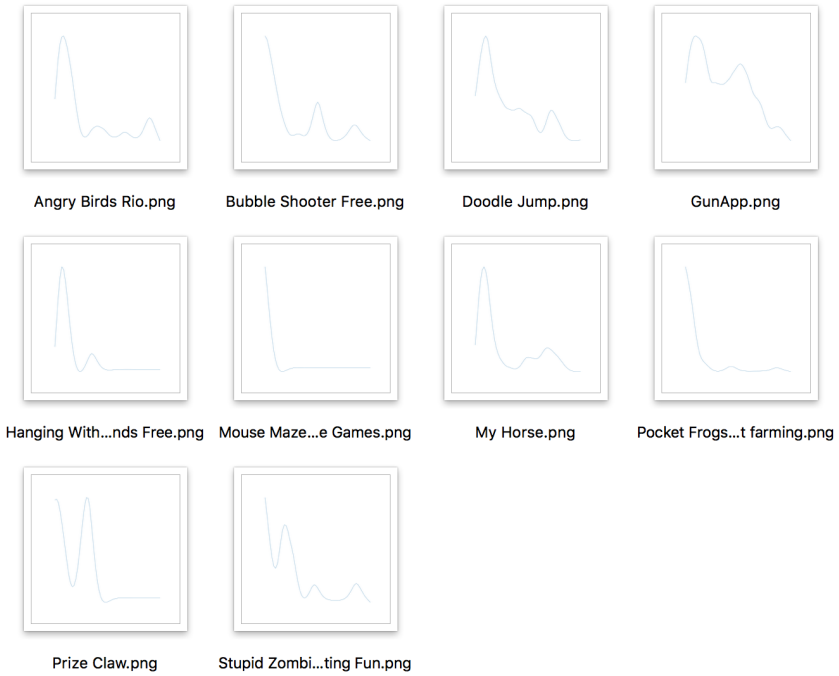

Figure A7.1. Curves in the subset 5 cluster 1

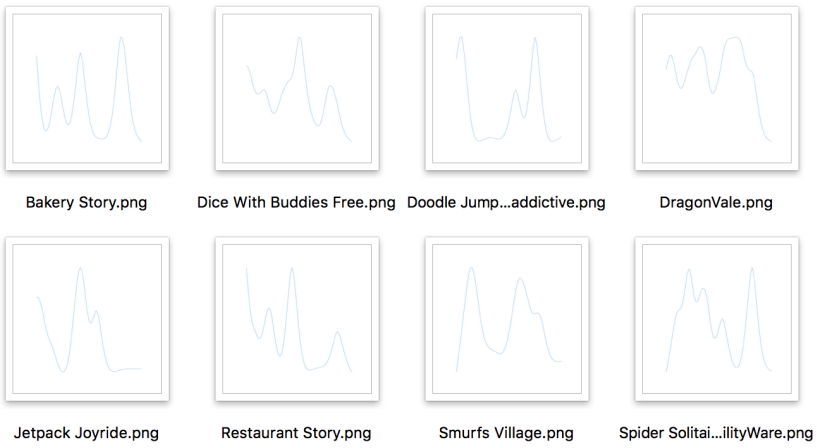

Figure A7.2. Curves in the subset 5 cluster 2

(continues)

Appendix 8. (continued)

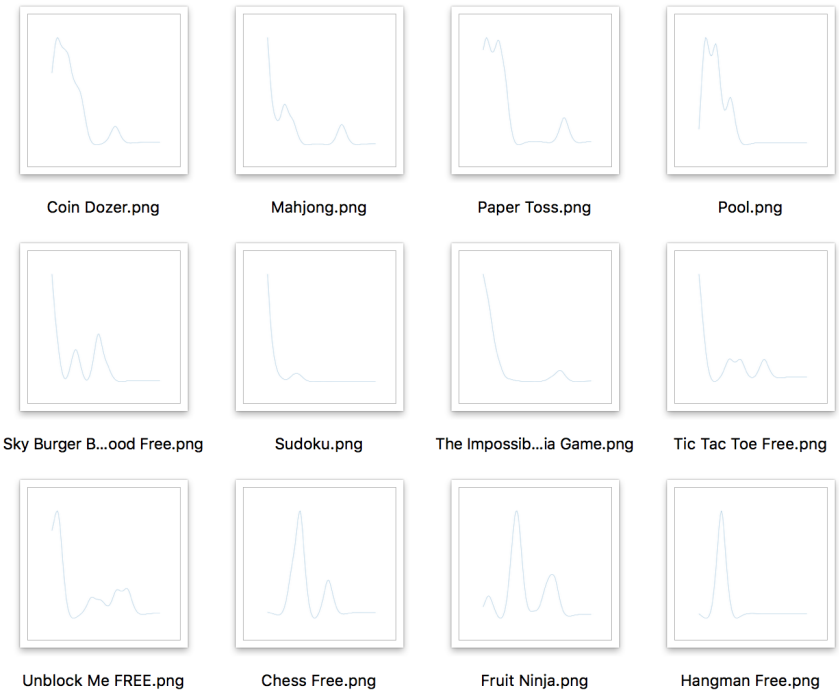

Figure A8.1. Curves in the subset 6 cluster 1

(continues)

Appendix 9. (continued)

Table A9.1. The links of mobile game

Application name	Application link
100 Balls	https://itunes.apple.com/app/id848218959?mt=8
100 Floors Can You Escape	https://itunes.apple.com/app/id517845106?mt=8
101 in 1 Games	https://itunes.apple.com/app/id458068537?mt=8
1010	https://itunes.apple.com/app/id911793120?mt=8
2048	https://itunes.apple.com/app/id840919914?mt=8
4 Pics 1 Word	https://itunes.apple.com/app/id595558452?mt=8
8 Ball Pool	https://itunes.apple.com/app/id543186831?mt=8
aa	https://itunes.apple.com/app/id905852173?mt=8
Agar io	https://itunes.apple.com/app/id995999703?mt=8
Angry Birds 2	https://itunes.apple.com/app/id880047117?mt=8
Angry Birds Go	https://itunes.apple.com/app/id642821482?mt=8
Angry Birds Rio	https://itunes.apple.com/app/id420635506?mt=8
Angry Birds Seasons	https://itunes.apple.com/app/id398157641?mt=8
Angry Birds Star Wars Free	https://itunes.apple.com/app/id557137623?mt=8
Angry Birds Star Wars II	https://itunes.apple.com/app/id645859810?mt=8
Angry Birds	https://itunes.apple.com/app/id343200656?mt=8
Angry Gran Run Running Game	https://itunes.apple.com/app/id563544551?mt=8
Ant Smasher Christmas a Free Game by the Best Cool Fun Games	https://itunes.apple.com/app/id488305689?mt=8
Asphalt 8 Airborne	https://itunes.apple.com/app/id610391947?mt=8
Atari s Greatest Hits	https://itunes.apple.com/app/id580765736?mt=8
Bakery Story	https://itunes.apple.com/app/id400506026?mt=8
Batman Arkham Origins	https://itunes.apple.com/app/id681370499?mt=8
Beat the Boss 2	https://itunes.apple.com/app/id572534490?mt=8
Beat the Boss	https://itunes.apple.com/app/id504656060?mt=8
Bejeweled Blitz	https://itunes.apple.com/app/id469960709?mt=8
Big Fish Casino Free Slots Vegas Slots Slot Tournaments Plus Poker Cards 21 and more	https://itunes.apple.com/app/id538212549?mt=8
Bike Race Free by Top Free Games	https://itunes.apple.com/app/id510461758?mt=8
Billionaire	https://itunes.apple.com/app/id881342787?mt=8
Bingo Bash with Wheel of Fortune Bingo and Slots games FREE	https://itunes.apple.com/app/id509098112?mt=8
Blackjack Free	https://itunes.apple.com/app/id460960682?mt=8
Blood Glory	https://itunes.apple.com/app/id466067444?mt=8
Boom Beach	https://itunes.apple.com/app/id672150402?mt=8
Bubble Mania	https://itunes.apple.com/app/id534342529?mt=8

(continues)

Appendix 9. (continued)

Table A9.1. The links of mobile game

Bubble Shooter Free	https://itunes.apple.com/app/id508066254?mt=8
Bubble Witch 2 Saga	https://itunes.apple.com/app/id834393815?mt=8
Campus Life Fashion Girl	https://itunes.apple.com/app/id504634395?mt=8
Can You Escape	https://itunes.apple.com/app/id658293394?mt=8
Candy Crush Saga	https://itunes.apple.com/app/id553834731?mt=8
Candy Crush Soda Saga	https://itunes.apple.com/app/id850417475?mt=8
Castle Clash Age of Legends	https://itunes.apple.com/app/id692669501?mt=8
Charades Free	https://itunes.apple.com/app/id653967729?mt=8
Checkers Free	https://itunes.apple.com/app/id294664915?mt=8
Chess Free	https://itunes.apple.com/app/id311395856?mt=8
Clash of Clans	https://itunes.apple.com/app/id529479190?mt=8
Club Penguin	https://itunes.apple.com/app/id505544063?mt=8
Clumsy Ninja	https://itunes.apple.com/app/id561416817?mt=8
Coin Dozer	https://itunes.apple.com/app/id372836496?mt=8
Contract Killer Zombies	https://itunes.apple.com/app/id438979179?mt=8
Contract Killer	https://itunes.apple.com/app/id451938707?mt=8
Cookie Jam	https://itunes.apple.com/app/id727296976?mt=8
Cooking Fever	https://itunes.apple.com/app/id714796093?mt=8
Covet Fashion The Game for Dresses Hairstyles and Shopping	https://itunes.apple.com/app/id620112416?mt=8
Criminal Case	https://itunes.apple.com/app/id767473889?mt=8
Crossy Road Endless Arcade Hopper	https://itunes.apple.com/app/id924373886?mt=8
CSR Classics	https://itunes.apple.com/app/id598603610?mt=8
CSR Racing	https://itunes.apple.com/app/id469369175?mt=8
DEAD TRIGGER	https://itunes.apple.com/app/id640111933?mt=8
Deal or No Deal	https://itunes.apple.com/app/id528829332?mt=8
Deer Hunter 2014	https://itunes.apple.com/app/id583222866?mt=8
Deer Hunter 2015	https://itunes.apple.com/app/id893372761?mt=8
Deer Hunter Challenge	https://itunes.apple.com/app/id394873600?mt=8
Deer Hunter Reloaded	https://itunes.apple.com/app/i101593d475?mt=8
Despicable Me Minion Rush	https://itunes.apple.com/app/id596402997?mt=8
Diamond Dash	https://itunes.apple.com/app/id461402734?mt=8
Dice With Buddies Free	https://itunes.apple.com/app/id432750508?mt=8
Doodle Jump FREE	https://itunes.apple.com/app/id456355158?mt=8
Doodle Jump	https://itunes.apple.com/app/id307727765?mt=8
Dots A Game About Connecting	https://itunes.apple.com/app/id632285588?mt=8
DoubleDown Casino Free Slots Video Poker Blackjack and More	https://itunes.apple.com/app/id485126024?mt=8
Drag Racing Classic	https://itunes.apple.com/app/id470867961?mt=8

(continues)

Appendix 9. (continued)

Table A9.1. The links of mobile game

Dragon City Mobile	https://itunes.apple.com/app/id591641526?mt=8
DragonVale	https://itunes.apple.com/app/id440045374?mt=8
Draw Something Free	https://itunes.apple.com/app/id488627858?mt=8
Dream League Soccer	https://itunes.apple.com/app/id472315002?mt=8
Dumb Ways to Die 2 The Games	https://itunes.apple.com/app/id929346489?mt=8
Dumb Ways to Die	https://itunes.apple.com/app/id639930688?mt=8
Episode Choose Your Story	https://itunes.apple.com/app/id656971078?mt=8
Evil Apples A Filthy Adult Card Party Game	https://itunes.apple.com/app/id645705454?mt=8
Exploration Lite	https://itunes.apple.com/app/id474737980?mt=8
Fair Food Maker Game Make Fair Foods and Play Free Carnival Games	https://itunes.apple.com/app/id525246631?mt=8
Fallout Shelter	https://itunes.apple.com/app/id991153141?mt=8
Family Guy The Quest for Stuff	https://itunes.apple.com/app/id838500730?mt=8
Farm Heroes Saga	https://itunes.apple.com/app/id608206510?mt=8
Farm Story	https://itunes.apple.com/app/id367107953?mt=8
FarmVille 2 Country Escape	https://itunes.apple.com/app/id824318267?mt=8
Fashion Story	https://itunes.apple.com/app/id420590864?mt=8
FIFA 15 Ultimate Team New Season	https://itunes.apple.com/app/id870794720?mt=8
Flappy Wings FREE	https://itunes.apple.com/app/id808176012?mt=8
Flow Free	https://itunes.apple.com/app/id526641427?mt=8
Frontline Commando	https://itunes.apple.com/app/id464131808?mt=8
Frozen Free Fall	https://itunes.apple.com/app/id692412579?mt=8
Fruit Ninja Free	https://itunes.apple.com/app/id403858572?mt=8
Fruit Ninja	https://itunes.apple.com/app/id362949845?mt=8
Fun Run 2 Multiplayer Race	https://itunes.apple.com/app/id920482331?mt=8
Fun Run Multiplayer Race	https://itunes.apple.com/app/id547201991?mt=8
Game of War Fire Age	https://itunes.apple.com/app/id667728512?mt=8
Geometry Dash Lite	https://itunes.apple.com/app/id698255242?mt=8
Glow Hockey 2 FREE	https://itunes.apple.com/app/id346453382?mt=8
GSN Casino FREE Slots Bingo Video Poker and Cards	https://itunes.apple.com/app/id469231420?mt=8
Guess The Emoji	https://itunes.apple.com/app/id698848120?mt=8
GunApp	https://itunes.apple.com/app/id303498337?mt=8
Hanging With Friends Free	https://itunes.apple.com/app/id440786655?mt=8
Hangman Free	https://itunes.apple.com/app/id327449554?mt=8
Happy Wheels	https://itunes.apple.com/app/id648668184?mt=8
Hardest Game Ever 2	https://itunes.apple.com/app/id606080169?mt=8
Hay Day	https://itunes.apple.com/app/id506627515?mt=8
Head Soccer	https://itunes.apple.com/app/id487119327?mt=8

(continues)

Appendix 9. (continued)

Table A9.1. The links of mobile game

Heads Up	https://itunes.apple.com/app/id623592465?mt=8
High School Story	https://itunes.apple.com/app/id631658837?mt=8
Highway Rider	https://itunes.apple.com/app/id494833223?mt=8
Hill Climb Racing	https://itunes.apple.com/app/id564540143?mt=8
Hit Tennis 3 Swipe flick the ball	https://itunes.apple.com/app/id506515857?mt=8
Home Design Story	https://itunes.apple.com/app/id547861844?mt=8
Hungry Shark Evolution	https://itunes.apple.com/app/id535500008?mt=8
Ice Age Village	https://itunes.apple.com/app/id467577200?mt=8
Icon Pop Quiz	https://itunes.apple.com/app/id568334356?mt=8
Injustice Gods Among Us	https://itunes.apple.com/app/id575658129?mt=8
Iron Man 3 The Official Game	https://itunes.apple.com/app/id593586999?mt=8
Jelly Jump	https://itunes.apple.com/app/id645949180?mt=8
Jelly Splash	https://itunes.apple.com/app/id645949180?mt=8
Jenga	https://itunes.apple.com/app/id392915994?mt=8
Jetpack Joyride	https://itunes.apple.com/app/id457446957?mt=8
Jewel Mania	https://itunes.apple.com/app/id561326449?mt=8
Jigsaw Puzzle	https://itunes.apple.com/app/id495583717?mt=8
Jurassic Park Builder	https://itunes.apple.com/app/id499250722?mt=8
Kids Doodle Movie Kids Color Draw	https://itunes.apple.com/app/id460712294?mt=8
Kill Shot	https://itunes.apple.com/app/id839703707?mt=8
Kim Kardashian Hollywood	https://itunes.apple.com/app/id860822992?mt=8
LEGO Star Wars The Complete Saga	https://itunes.apple.com/app/id727420266?mt=8
Little Dentist kids games game for kids	https://itunes.apple.com/app/id643552439?mt=8
Logos Quiz Guess the logos	https://itunes.apple.com/app/id949241585?mt=8
MADDEN NFL Mobile	https://itunes.apple.com/app/id855627886?mt=8
Magic Jigsaw Puzzles	https://itunes.apple.com/app/id439873467?mt=8
Mahjong	https://itunes.apple.com/app/id347878114?mt=8
Make It Rain The Love of Money	https://itunes.apple.com/app/id839599050?mt=8
Mancala FS5 FREE	https://itunes.apple.com/app/id300868307?mt=8
MARVEL Contest of Champions	https://itunes.apple.com/app/id896112560?mt=8
Megapolis	https://itunes.apple.com/app/id580765736?mt=8
MetalStorm Online	https://itunes.apple.com/app/id425746946?mt=8
Minecraft Pocket Edition	https://itunes.apple.com/app/id479516143?mt=8
Mini Golf MatchUp	https://itunes.apple.com/app/id577650466?mt=8
Modern War	https://itunes.apple.com/app/id468327549?mt=8
MORTAL KOMBAT X	https://itunes.apple.com/app/id949701151?mt=8
Mouse Maze Free Game by Top Free Games	https://itunes.apple.com/app/id464837515?mt=8
Mr Jump	https://itunes.apple.com/app/id955157084?mt=8

(continues)

Appendix 9. (continued)

Table A9.1. The links of mobile game

My Boo Virtual Pet with Mini Games for Kids Boys and Girls	https://itunes.apple.com/app/id706099830?mt=8
My Horse	https://itunes.apple.com/app/id421167112?mt=8
My Singing Monsters	https://itunes.apple.com/app/id521671873?mt=8
My Talking Angela	https://itunes.apple.com/app/id909351158?mt=8
My Talking Tom	https://itunes.apple.com/app/id657500465?mt=8
myVEGAS Slots Play Free Las Vegas Casino Games Slot Machines and Spin Win	https://itunes.apple.com/app/id714508224?mt=8
NBA JAM by EA SPORTS LITE	https://itunes.apple.com/app/id431761169?mt=8
Need for Speed No Limits	https://itunes.apple.com/app/id883393043?mt=8
NFL Pro 2014 The Ultimate Football Simulation	https://itunes.apple.com/app/id518244464?mt=8
NinJump	https://itunes.apple.com/app/id379471852?mt=8
Office Jerk	https://itunes.apple.com/app/id423593206?mt=8
PAC MAN Lite	https://itunes.apple.com/app/id293778748?mt=8
Panda Pop Bubble Shooter	https://itunes.apple.com/app/id700970012?mt=8
Paper Toss 2 0	https://itunes.apple.com/app/id477846014?mt=8
Paper Toss	https://itunes.apple.com/app/id317917431?mt=8
Parking Mania Free	https://itunes.apple.com/app/id466260931?mt=8
Pet Rescue Saga	https://itunes.apple.com/app/id572821456?mt=8
Pet Shop Story	https://itunes.apple.com/app/id481890182?mt=8
Piano Tiles Don t Tap The White Tile	https://itunes.apple.com/app/id848160327?mt=8
Pixel Gun 3D	https://itunes.apple.com/app/id640111933?mt=8
Plants vs Zombies 2	https://itunes.apple.com/app/id597986893?mt=8
Plumber Crack	https://itunes.apple.com/app/id470225328?mt=8
Pocket Frogs Free pet farming	https://itunes.apple.com/app/id386644958?mt=8
Pool	https://itunes.apple.com/app/id332184886?mt=8
Pop the Lock	https://itunes.apple.com/app/id979100082?mt=8
Prize Claw	https://itunes.apple.com/app/id434800417?mt=8
QuizUp	https://itunes.apple.com/app/id718421443?mt=8
Racing Rivals	https://itunes.apple.com/app/id604186826?mt=8
Real Basketball	https://itunes.apple.com/app/id597855590?mt=8
Real Racing 3	https://itunes.apple.com/app/id556164008?mt=8
Restaurant Story	https://itunes.apple.com/app/id394807905?mt=8
Robot Unicorn Attack 2	https://itunes.apple.com/app/id541672969?mt=8
Ruzzle	https://itunes.apple.com/app/id504265646?mt=8
SCRABBLE Free	https://itunes.apple.com/app/id501724085?mt=8
Shooty Skies Endless Arcade Flyer	https://itunes.apple.com/app/id962993853?mt=8
SimCity BuildIt	https://itunes.apple.com/app/id913292932?mt=8
Sky Burger Build Match Food Free	https://itunes.apple.com/app/id311972587?mt=8

(continues)

Appendix 9. (continued)

Table A9.1. The links of mobile game

Slotomania Casino Las Vegas Free Slot Machine Games Bet Spin Win	https://itunes.apple.com/app/id447553564?mt=8
Slots Pharaoh s Way The best free casino slots and slot tournaments	https://itunes.apple.com/app/id522408559?mt=8
Smash Hit	https://itunes.apple.com/app/id603527166?mt=8
Smashy Road Wanted	https://itunes.apple.com/app/id1020119327?mt=8
Smurfs Village	https://itunes.apple.com/app/id399648212?mt=8
Sniper 3D Assassin Shoot to Kill by Fun Games For Free	https://itunes.apple.com/app/id930574573?mt=8
Sniper Shooter by Fun Games for Free	https://itunes.apple.com/app/id635573390?mt=8
Social Girl	https://itunes.apple.com/app/466959933?mt=8
Solitaire	https://itunes.apple.com/app/id500963785?mt=8
SongPop	https://itunes.apple.com/app/id518042655?mt=8
Sonic Dash	https://itunes.apple.com/app/id582654048?mt=8
Spider Solitaire Free by MobilityWare	https://itunes.apple.com/app/id395979574?mt=8
SpongeBob Diner Dash	https://itunes.apple.com/app/id333578095?mt=8
Stick Hero	https://itunes.apple.com/app/id918338898?mt=8
Stickman BMX Free	https://itunes.apple.com/app/id443335576?mt=8
Stickman Skater Free	https://itunes.apple.com/app/id368670751?mt=8
Stupid Zombies Free Gun Shooting Fun	https://itunes.apple.com/app/id412453843?mt=8
Subway Surfers	https://itunes.apple.com/app/id512939461?mt=8
Sudoku	https://itunes.apple.com/app/id366247306?mt=8
Tap Pet Hotel	https://itunes.apple.com/app/id415569164?mt=8
Temple Run 2	https://itunes.apple.com/app/id572395608?mt=8
Temple Run	https://itunes.apple.com/app/id420009108?mt=8
Tetris Blitz	https://itunes.apple.com/app/id632827808?mt=8
The Blockheads	https://itunes.apple.com/app/id519596406?mt=8
THE GAME OF LIFE Classic Edition	https://itunes.apple.com/app/id326912270?mt=8
The Impossible Test Fun Free Trivia Game	https://itunes.apple.com/app/id325955444?mt=8
The Simpsons Tapped Out	https://itunes.apple.com/app/id497595276?mt=8
The Sims FreePlay	https://itunes.apple.com/app/id466965151?mt=8
The Walking Dead No Man s Land	https://itunes.apple.com/app/id970417047?mt=8
Tic Tac Toe Free	https://itunes.apple.com/app/id289278457?mt=8
Tiny Tower Free City Building	https://itunes.apple.com/app/id422667065?mt=8
Tiny Wings	https://itunes.apple.com/app/id417817520?mt=8
Toilet Time Mini Games to Play in the Bathroom	https://itunes.apple.com/app/id736787261?mt=8
Traffic Racer	https://itunes.apple.com/app/id547101139?mt=8
Trigger Fist	https://itunes.apple.com/app/id531184261?mt=8
Trivia Crack	https://itunes.apple.com/app/id651510680?mt=8

(continues)

Appendix 9. (continued)

Table A9.1. The links of mobile game

Turbo FAST	https://itunes.apple.com/app/id604752655?mt=8
Two Dots	https://itunes.apple.com/app/id880178264?mt=8
Unblock Me FREE	https://itunes.apple.com/app/id315019111?mt=8
UNO Friends The Classic Card Game Goes Social	https://itunes.apple.com/app/id537263603?mt=8
Unroll Me unblock the slots	https://itunes.apple.com/app/id769574372?mt=8
Waterslide Extreme	https://itunes.apple.com/app/id322410766?mt=8
What s the Difference spot the differences in this photo hunt puzzle of hidden object games	https://itunes.apple.com/app/id820452091?mt=8
What s the Word new quiz with pics and word	https://itunes.apple.com/app/id573511269?mt=8
Where s My Perry Free	https://itunes.apple.com/app/id949701151?mt=8
Where s My Water 2	https://itunes.apple.com/app/id638853147?mt=8
Where s My Water Free	https://itunes.apple.com/app/id467810884?mt=8
Where s My Water	https://itunes.apple.com/app/id449735650?mt=8
White Tiles 4 Piano Master Don t Touch the White Tile and Trivia games Free	https://itunes.apple.com/app/id1146128499?mt=8
Word Search Puzzles	https://itunes.apple.com/app/id609067187?mt=8
Word Streak With Friends	https://itunes.apple.com/app/?mt=8
WordBrain	https://itunes.apple.com/app/id708600202?mt=8
Words With Friends	https://itunes.apple.com/app/id804379658?mt=8
ZigZag	https://itunes.apple.com/app/id951364656?mt=8
Zombie Highway	https://itunes.apple.com/app/id376412160?mt=8
Zynga Poker Texas Holdem	https://itunes.apple.com/app/id354902315?mt=8

(continues)